IBM 大型主机系列

大型主机DB2数据库基础教程

张　颖　李六旬　**主　编**

王亚娟　孙佩锋　孙　哲　易　蕾　吴春利

莫李华　姚文杰　黎春阳　孙　彬　邹　为　**编　著**

電子工業出版社

Publishing House of Electronics Industry

北京・BEIJING

内 容 简 介

本书针对目前 IBM 大型主机 DB2 数据库知识专业性强、相关中文书籍很少、英文技术手册可读性欠佳等问题，以语言通俗为原则，以实践应用为目的，全面讲述了主机 DB2 的基础、Data Sharing 理论、DB2 常用的 Utility、DB2 常用命令，以及 DB2 系统维护等方面的知识，并概要介绍了常用的几种 DB2 工具的使用方法。同时在每个章节后配有针对性的案例、习题与答案，方便读者对知识的深入理解和巩固。

本书的编著融入了大型主机项目研发和系统维护实践过程的丰富知识与经验，可作为已开展主机专业课程的高校教学资料；对于已从事主机 DB2 数据库专业相关工作的技术人员，甚至科研人员，也是一本不可多得的参考书。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

大型主机 DB2 数据库基础教程 / 张颖，李六旬主编；王亚娟等编著. —北京：电子工业出版社，2010.7
（IBM 大型主机系列）
ISBN 978-7-121-11058-0

Ⅰ. ①大… Ⅱ. ①张… ②李… ③王… Ⅲ. ①大型计算机－教材 Ⅳ. ①TP338.4

中国版本图书馆 CIP 数据核字（2010）第 106636 号

责任编辑：高洪霞
印　　刷：北京智力达印刷有限公司
装　　订：北京中新伟业印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×980　1/16　印张：22.5　字数：471 千字
印　　次：2010 年 7 月第 1 次印刷
印　　数：3000 册　定价：80.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

序　言

IBM 大型主机遍布全球众多大型企业，广泛应用于银行、证券、保险、电信、航空等行业。时至今日，大型主机在 IT 界仍扮演着极其重要的角色。仅从银行业来看，全球最大的 500 家银行中，绝大多数都采用大型主机来搭建核心信息系统。

大型主机有别于我们常见的个人电脑、服务器或小型机，其操作系统 z/OS 也与常见的 Windows、Linux 或 UNIX 操作系统大不相同，运行在其上的中间件、数据库软件也有其独特性。由于大型主机专业性很强，且通常只在大型企业中应用，全球的大型主机专业技术人才总量较小。据统计，目前全球专业的大型主机专业技术人员约 10 万人，且其中相当一部分从 20 世纪 60 年代大型主机诞生之时起就开始从事大型主机的技术工作，新生代的技术人员占比并不大。此外，该领域的技术人才培养难度较大，常常依赖传统的师傅带徒弟方式，缺乏系统的培训体系和教材，培养周期较长，一般需要 5 到 10 年的不懈努力和经验积累，才能成为大型主机某个领域的专家，培训成本相对也较高，常常无法满足多数大型主机企业用户甚至IBM公司自身对专业人才培养的需求。而国内大型银行近十年发展迅速，对开发和管理核心银行系统的大型主机专业技术人才需求旺盛，大型主机人才常常供不应求，对此银行管理者感受尤为深刻。一个基本事实就是，大型主机的基础培训教材匮乏，现有教材内容相对陈旧，无法反映大型主机软硬件技术、应用推广技术和维护管理技术的发展现状。

中国工商银行是中国金融行业电子化建设的先行者，在大型主机的开发和应用方面已积累了 20 多年的经验。早在 1999 年，中国工商银行就在国内率先启动了建设超大规模数据处理中心的“9991”数据大集中工程；搭建了以大型主机系统为核心的银行业务处理平台；实现了关键业务数据和业务处理的大集中。数据中心（北京）也在 2000 年应运而生，并在 10 年间迅速发展壮大，向着世界一流的大型数据中心迈进。一直以来，数据中心（北京）非常重视人才培养工作，致力于打造卓越的学习型团队和学习型组织。经过多年的不懈努力，目前中心已经建立起较为完善的专业化人才培养体系，营造出技术人才快速成长的环境，大型主机专业人才培养则是其中一个重要的组成部分。我们不断地挖掘大型主机

课程内容的深度和广度，自行组织编制了一系列培训课件和教材，在每年的新员工培训和专家人才培养方面发挥了显著效果。一路走来，一支专业化、高素质的工商银行大型主机专业团队已经形成！

本书的作者正是中国工商银行数据中心（北京）系统部经验丰富的工程师们。这是一支朝气蓬勃的团队，深刻了解中国用户在大型主机知识和技能方面的需求。他们站在工行信息科技前辈的肩膀上，系统地整理了大型主机在金融行业应用开发、管理方面的丰富知识和宝贵经验，精心汇集了工商银行在大型主机领域近 20 年潜心耕耘的成果，大量参阅了各类英文技术文献，在百忙工作之余投入大量时间和精力著成系列丛书。希望该书能让国内同行在大型机知识和技能培训方面得所借鉴，让每一个对 IBM Z 系列大型机感兴趣的读者有所收获。

作为中国工商银行数据中心（北京）的总经理，我衷心希望能借助本套系列基础教程的推出，搭建起我们与国内同行之间的沟通平台，开启我们与读者之间的交流之门，在为国内同行和广大读者提供大型主机的技术参考的同时，还能够收到读者反馈回来的宝贵意见和建议，以实现沟通交流、互补互馈，力争把该系列教程做得精益求精。希望这本中文教材能够成为您的良师益友，能为您今后的工作和学习贡献一份力量！



中国工商银行数据中心（北京） 总经理

2010 年 4 月于北京

前　言　Preface

DB2 是 IBM 公司数据管理产品线上最知名也最成功的产品，在许多行业尤其是银行业得到非常广泛的应用，是主流数据库管理系统。DB2 产品适用于多种操作系统平台。对于运行于大型主机环境下的 DB2，由于专业性非常强，相关中文书籍很少，英文技术手册的可读性较差，缺乏系统性介绍主机 DB2 的辅导教程，从事主机 DB2 工作的人员很难利用目前市面上已有的书籍系统全面地掌握主机 DB2 的相关技术。为了帮助主机数据库工作的人员更轻松快速地入门，为从事数据库相关工作的同事提供一份较全面、实用的参考教材，主机 DB2 专业团队编写了本教程。本书力争用通俗的语言，由浅入深地介绍主机 DB2 的基础、Data Sharing 理论、常用的 Utility、常用命令，以及 DB2 系统维护等方面的知识，同时概要地介绍几种常用的 DB2 工具的使用方法。

除主编作者外，参与编著工作的还有中国工商银行数据中心（北京）系统部 DB2 专业团队王亚娟、孙佩锋、易蕾、孙哲、吴春利、莫李华、姚文杰、黎春阳、孙彬、邹为等同志，不仅对主机 DB2 产品进行了全面、深入的介绍，还融入了他们多年从事主机数据库研发和日常维护工作中积累的丰富知识和经验。书中每一章节后的习题和案例，是该团队所有成员在日常工作培训中常见和实际使用的，相信会对有志于从事主机数据库工作的同事提供帮助，使入门者更轻松快速地入门。

本书由这些同志利用业余时间倾注大量心血编著而成，所有文稿都经过了反复推敲和修订，旨在为从事主机 DB2 研发、测试、维护等工作的同志们提供一份较为全面、实用的参考资料。由于能力水平有限，本书难免存在不足和纰漏，还请读者批评指正，以求不断完善。

目　录　Contents

第 1 章　数据库理论基础　1

本章为数据库理论基础，主要介绍数据库的基本概念，内容包括数据库技术发展、数据库系统特点、关系型数据库基本概念、SQL 语言和关系型数据库设计方法。

1.1　数据库基本概念……1

1.1.1　数据库技术的产生和发展……1

1.1.2　数据库系统的特点……2

1.1.3　数据模型……4

1.2　关系型数据库基本概念……7

1.2.1　基本术语……7

1.2.2　Codd 准则……7

1.2.3　关系完整性……9

1.3　SQL 语言概述……9

1.3.1　SQL 语言分类……10

1.3.2　数据类型……10

1.3.3　数据控制语言（DCL）……13

1.3.4　数据定义语言（DDL）……14

1.3.5　数据操作语言（DML）……16

1.3.6　SQL 函数……24

1.3.7　制定约束……27

1.3.8　静态 SQL 与动态 SQL……29

1.4　关系数据库设计……30

1.4.1　关系规范化……30

1.4.2　实体——关系模型……33

1.4.3　关系数据库设计的基本步骤……34

1.5 课后习题 44

第 2 章 主机 DB2 基础 45

本章为主机 DB2 基础，主要介绍 DB2 产品的基本架构，包括 DB2 产品发展概述、DB2 数据基本结构、DB2 系统结构组成、DB2 系统运行环境、DB2 并发控制机制、应用程序的管理和开发、数据库安全控制等。通过对以上知识点的讲解，让读者能够对 DB2 系统的基础知识有一定的理解。

2.1 DB2 产品发展概述 45

2.2 DB2 数据库对象 47

2.2.1 DB2 数据库对象概貌 47

2.2.2 Database 48

2.2.3 Storage Group 48

2.2.4 Tablespace 49

2.2.5 Table 55

2.2.6 Index 56

2.2.7 View 59

2.2.8 Synomas 61

2.2.9 Alias 61

2.2.10 Trigger 62

2.2.11 Store Procedure 62

2.2.12 创建数据库对象的方法 63

2.2.13 OWNER 的概念 63

2.2.14 数据库对象的命名规范 64

2.2.15 数据库对象对应 VSAM 数据集的命名规范 64

2.2.16 查询数据库对象的方法 65

2.3 DB2 系统结构组成 65

2.3.1 DB2 系统结构概貌 66

2.3.2 DB2 Catalog 66

2.3.3 DB2 Directory 67

2.3.4 默认数据库 69

2.3.5 work file database……69
2.3.6 Active and Archive log……69
2.3.7 Bootstrap Data Set（BSDS）……70
2.3.8 Buffer pool……71
2.3.9 EDM pool……71
2.3.10 RID pool……72
2.3.11 DSNZPARM……72
2.4 DB2 系统运行环境……73
2.4.1 DB2 系统的地址空间……73
2.4.2 DB2 Attachment Facilities……74
2.4.3 DB2 与分布式数据……77
2.4.4 DB2 与 z/OS……78
2.4.5 DB2 与 Parallel Sysplex……78
2.4.6 DB2 与安全服务……79
2.4.7 DB2 与 DFSMS……79
2.4.8 DB2 与 WLM……79
2.5 DB2 并发控制机制……81
2.5.1 数据一致性……81
2.5.2 DB2 事务的概念……82
2.5.3 COMMIT 和 ROLLBACK 操作的结果……83
2.5.4 不成功的事务的结果……83
2.5.5 事务隔离级别……84
2.5.6 锁机制……86
2.5.7 锁的挂起、超时和死锁……89
2.6 DB2 应用程序管理……91
2.6.1 基本概念……91
2.6.2 应用程序的准备过程……92
2.6.3 存储过程……94
2.6.4 UDF……98

2.6.5 触发器……99
2.7 DB2 应用程序 SQL 优化……100
2.7.1 应用程序编写的通用原则……100
2.7.2 编写高效的谓词……103
2.7.3 DB2 谓词管理……124
2.7.4 高效地使用宿主变量……127
2.7.5 编写高效的子查询……129
2.7.6 如何判断一个 SQL 有问题……134
2.7.7 小结……135
2.8 DB2 安全控制与审计……135
2.8.1 数据库安全控制范围……135
2.8.2 DB2 安全控制简介……136
2.8.3 DB2 对象访问控制……138
2.8.4 DB2 审计……144
2.9 课后习题……148
第 3 章 DB2 Data Sharing 基础 150
本章主要介绍 DB2 Data Sharing 方面的知识，内容包括 Data Sharing 特性介绍、与 Stand Alone 的区别、Data Sharing 与 SYSPLEX 的关系、Data Sharing 系统架构、Data Sharing 的实现和恢复等，帮助读者掌握 Data Sharing 的基本概念和基本使用方法。
3.1 DB2 Data Sharing 介绍……150
3.1.1 Parallel Sysplex 与 DB2 Data Sharing 简介……151
3.1.2 DB2 Data Sharing 技术的优势……153
3.2 DB2 Data Sharing 体系架构……156
3.2.1 DB2 Data Sharing 架构的问题及解决方法……157
3.2.2 并发性和数据一致性控制……158
3.2.3 DB2 Data Sharing 的连续可用性……163
3.2.4 异常情况对 DB2 Data Sharing 的可用性影响……165
3.3 DB2 Data Sharing 的实现……166
3.3.1 命名规则……166

3.3.2 DB2 日志 170
3.3.3 DB2 Data Sharing 重要参数 171
3.4 DB2 Data Sharing 的恢复 172
3.4.1 对某个 DB2 表的恢复 172
3.4.2 对某些组件异常的恢复 174
3.4.3 Sysplex Failure Management（SFM） 175
3.4.4 Automatic Restart Management（ARM） 175
3.4.5 DB2 MEMBER Light 模式重启 175
3.5 课后习题 176
第 4 章 DB2 常用 Utility 基础 177
本章为 DB2 常用 Utility 基础，主要介绍 DB2 产品自带 Utility 方面的知识，内容涉及 Utility 的调用方法、Online Utility 和 Offline Utility 的介绍等。
4.1 DB2 Utility 简介 177
4.1.1 什么是 Utility 177
4.1.2 Utility 的分类 177
4.1.3 调用 Utility 的方法 178
4.2 DB2 Online Utility 178
4.2.1 如何调用 DB2 Online Utility 178
4.2.2 BACKUP SYSTEM Utility 179
4.2.3 COPY Utility 180
4.2.4 LISTDEF Utility 184
4.2.5 LOAD Utility 187
4.2.6 REBUILD INDEX Utility 195
4.2.7 RECOVER Utility 198
4.2.8 REORG TABLESPACE Utility 202
4.2.9 REPAIR Utility 211
4.2.10 RUNSTATS Utility 213
4.2.11 TEMPLATE Utility 218
4.2.12 UNLOAD Utility 221

4.2.13 监控与控制 DB2 Online Utility ····· 225
4.3 DB2 Stand Alone Utility ····· 226
4.3.1 如何调用 Stand Alone Utility ····· 226
4.3.2 DSN1COPY Utility ····· 226
4.3.3 DSN1LOGP Utility ····· 231
4.3.4 DSN1PRNT Utility ····· 234
4.4 课后习题 ····· 238
第 5 章 DB2 常用命令 239
本章主要介绍 DB2 的基本命令，涉及启下系统、检查 DB2 运行状态、检查数据库对象状态、解决异常状态等操作。
5.1 DB2 命令介绍 ····· 239
5.1.1 DB2 命令的作用范围 ····· 239
5.1.2 DB2 命令的提交方式 ····· 240
5.1.3 DB2 命令的使用方法 ····· 240
5.2 DB2 系统相关命令 ····· 241
5.2.1 –START DB2 ····· 241
5.2.2 –STOP DB2 ····· 242
5.2.3 –SET SYSPARM ····· 242
5.2.4 –DISPLAY GROUP ····· 243
5.2.5 –RECOVER BSDS ····· 244
5.2.6 –RECOVER INDOUBT ····· 244
5.3 DATABASE 相关命令 ····· 245
5.3.1 –ACCESS DATABASE ····· 245
5.3.2 –START DATABASE ····· 245
5.3.3 –STOP DATABASE ····· 247
5.3.4 –DISPLAY DATABASE ····· 247
5.4 BP 和 GBP 相关命令 ····· 250
5.4.1 –ALTER BUFFERPOOL ····· 250
5.4.2 –ALTER GROUPBUFFERPOOL ····· 252

5.4.3 –DISPLAY BUFFERPOOL ····· 253
5.4.4 –DISPLAY GROUPBUFFERPOOL ····· 255
5.5 Utility 相关命令 ····· 256
5.5.1 –ALTER Utility ····· 256
5.5.2 –DISPLAY Utility ····· 257
5.5.3 –TERM Utility ····· 257
5.6 TRACE 相关命令 ····· 258
5.6.1 –START TRACE ····· 258
5.6.2 –STOP TRACE ····· 260
5.6.3 –DISPLAY TRACE ····· 261
5.7 PROCEDURE 相关命令 ····· 262
5.7.1 –DISPLAY PROCEDURE ····· 262
5.7.2 –START PROCEDURE ····· 263
5.7.3 –STOP PROCEDURE ····· 264
5.8 DDF 相关命令 ····· 265
5.8.1 –START DDF ····· 265
5.8.2 –STOP DDF ····· 266
5.8.3 –DISPLAY DDF ····· 267
5.9 LOG 相关命令 ····· 267
5.9.1 –ARCHIVE LOG ····· 267
5.9.2 –DISPLAY LOG ····· 269
5.10 THREAD 相关命令 ····· 270
5.10.1 –CANCEL THREAD ····· 270
5.10.2 –DISPLAY THREAD ····· 270
5.11 IRLM 相关命令 ····· 272
5.11.1 –START irlmproc ····· 272
5.11.2 –STOP irlmproc ····· 275
5.11.3 –TRACE CT ····· 275
5.11.4 –MODIFY irlmproc, ABEND ····· 276

5.11.5 –MODIFY irlmproc, DIAG ····· 276
5.11.6 –MODIFY irlmproc, PURGE ····· 277
5.11.7 –MODIFY irlmproc, SET ····· 278
5.11.8 –MODIFY irlmproc, STATUS ····· 279
5.12 课后习题 ····· 281

第 6 章 DB2 系统维护概述 283

本章为 DB2 系统维护方法介绍，主要包括系统备份和恢复、日常监控和健康检查、例行重组和性能分析调优等内容，旨在帮忙读者对 DB2 维护工作有较为全面的了解。

6.1 DB2 备份和恢复 ····· 283
6.1.1 DB2 备份及恢复的原则 ····· 283
6.1.2 如何制定最佳的备份策略 ····· 284
6.1.3 DB2 备份常用工具及使用方法 ····· 285
6.1.4 DB2 恢复工具及使用方法 ····· 286
6.1.5 DB2 数据库备份策略参考 ····· 289
6.2 DB2 日常健康检查 ····· 290
6.2.1 DB2 系统级检查 ····· 291
6.2.2 数据可用性和应用程序检查 ····· 297
6.3 DB2 重组 ····· 300
6.3.1 进行 DB2 重组的目的 ····· 300
6.3.2 DB2 REORG 的对象 ····· 301
6.3.3 DB2 重组的条件 ····· 301
6.3.4 DB2 重组的方法 ····· 302
6.3.5 DB2 重组的注意事项 ····· 304
6.3.6 DB2 重组常见问题及解决方法 ····· 305
6.4 DB2 RUNSTATS ····· 308
6.4.1 定期执行 RUNSTATS 的目的 ····· 308
6.4.2 何时需要进行 RUNSTATS 操作 ····· 308
6.4.3 RUNSTATS 注意点 ····· 309
6.5 STOSPACE ····· 309

6.6 DB2 ROTATE 操作……310
6.7 DB2 性能监控及调整……310
6.7.1 DB2 性能监控……310
6.7.2 DB2 性能调整……320
6.8 课后习题……323
附录 A DB2 Admin Tool 简介 325
附录 B DB2 PM 简介 329
附录 C SPUFI 简介 332
附录 D 参考答案 334
附录 E 常见主机资料缩语表 341
参考文献 344

第 1 章　数据库理论基础

本章为数据库理论基础，主要介绍数据库的基本概念，内容包括数据库技术发展、数据库系统特点、关系型数据库基本概念、SQL 语言和关系型数据库设计方法。



1.1　数据库基本概念

1.1.1　数据库技术的产生和发展

数据库技术是应数据管理任务的需要而产生的。数据的处理是指对各种数据进行收集、存储、加工和传播的一系列活动的总和。数据管理则是指对数据进行分类、组织、编码、存储、检索和维护，它是数据处理的中心问题。随着计算机硬件、软件技术的发展，在用户需求的推动下，数据管理技术经历了人工管理、文件系统、数据库系统三个阶段。

1．人工管理阶段

20 世纪 50 年代中期以前，计算机主要用于科学计算。当时的硬件状况是外存只有纸带、卡片、磁带，没有磁盘等直接存取的存储设备；软件状况是没有操作系统，没有管理数据的软件；数据处理方式是批处理。人工管理数据具有如下特点：

- 数据不保存；
- 应用程序管理数据；

- 数据不共享；
- 数据不具有独立性。

2．文件系统阶段

20 世纪 50 年代后期到 60 年代中期，这时硬件方面已有了磁盘、磁鼓等直接存取存储设备；软件方面，操作系统中已经有了专门的数据管理软件，一般称为文件系统；处理方式上不仅有了批处理，而且能够联机实时处理。文件系统管理数据具有如下特点：

- 数据可以长期保存；
- 由文件系统管理数据；
- 数据共享性差，冗余度大；
- 数据独立性差。

3．数据库系统阶段

20 世纪 60 年代后期以来，计算机用于管理的规模越来越大，应用越来越广泛，数据量急剧增长，同时多种应用、多种语言相互覆盖的共享数据集合的要求越来越强烈。这时硬件已有大容量磁盘，硬件价格下降；软件则价格上升，为编制和维护系统软件及应用程序所需的成本相对增加；在处理方式上，联机实时处理要求更多，并开始提出和考虑分布处理。

在这种背景下，以文件系统作为数据管理手段已经不能满足应用的需求，于是为解决多用户、多应用共享数据的需求，使数据为尽可能多的应用服务，数据库技术便应运而生，出现了统一管理数据的专门软件系统——数据库管理系统。用数据库系统来管理数据比文件系统具有明显的优点，从文件系统到数据库系统，标志着数据管理技术的飞跃。

1.1.2 数据库系统的特点

随着计算机技术的不断发展，数据处理技术也进入到数据库系统阶段，数据库技术在数据管理方面表现出强大的功能性和突出的优点。

1．数据结构化

数据结构化是数据库与文件系统的根本区别。在数据库系统中，数据不再针对某一应用，而是面向全组织，具有整体的结构化，不仅数据是结构化的，而且存取数据的方式也很灵活，可以存取数据库中的某一个数据项、一组数据项、一个记录或一组记录。而在文

件系统中，数据的最小存取单位是记录，颗粒度不能细到数据项。

2．数据的共享性高，冗余度低，易扩展

数据库系统从整体角度看待和描述数据，数据不再面向某个应用，而是面向整个系统，因此数据可以被多个用户、多个应用共享使用。数据共享可以大大减少数据冗余，节约存储空间。数据共享还能避免数据之间的不相容性与不一致性。

所谓数据的不一致性，是指同一数据不同拷贝的值不一样。采用人工管理或文件系统管理时，由于数据被重复存储，当被不同的应用使用和修改不同的拷贝时就很容易造成数据的不一致。在数据库中进行数据共享，减少了由于数据冗余而造成的不一致现象。

由于数据面向整个系统，是有结构的数据，不仅可以被多个应用共享使用，而且容易增加新的应用，这就使得数据库系统弹性大，易于扩充，可以适应各种用户的要求。可以取整体数据的各种子集用于不同的应用系统，当应用需求改变或增加时，只要重新选取不同的子集或加上一部分数据便可以满足新的需求。

3．数据独立性高

数据独立性是数据库领域中的一个常用术语，包括数据的物理独立性和数据的逻辑独立性。

物理独立性是指用户的应用程序与存储在磁盘上的数据库中的数据是相互独立的。也就是说，数据在磁盘上的数据库中怎样存储是由 DBMS（Database Management System，数据库管理系统）管理的，用户程序不需要了解，应用程序要处理的只是数据的逻辑结构，这样当数据的物理存储改变时，应用程序不用改变。

逻辑独立性是指用户的应用程序与数据库的逻辑结构是相互独立的，也就是说，数据的逻辑结构改变了，用户程序可以不变。数据与程序的独立，把数据的定义从程序中分离出去，加上数据的存取又由 DBMS 负责，从而简化了应用程序的编制，大大减少了应用程序的维护和修改。

4．数据由DBMS统一管理和控制

数据库的共享是并发共享，即多个用户可以同时存取数据库中的数据，甚至可以同时存取数据库中的同一个数据。为此，DBMS 还必须提供以下几方面的数据控制功能：

（1）数据的安全性保护

数据的安全性保护是指保护数据以防止不合法的使用造成的数据泄密和破坏，使每个

用户只能按规定，对某些数据以某些方式进行使用和处理。

（2）数据的完整性检查

数据的完整性检查是指检查数据的正确性、有效性和相容性。完整性检查将数据控制在有效的范围内，或保证数据之间满足一定的关系。

（3）并发控制

当多个用户的并发进程同时存取、修改数据库时，可能会发生相互干扰而得到错误的结果或使得数据库的完整性遭到破坏，因此必须对多用户的并发操作加以控制和协调。

（4）数据库恢复

计算机系统的硬件故障、软件故障、操作员的失误及故意的破坏也会影响数据库中数据的正确性，甚至造成数据库部分或全部数据的丢失。DBMS 必须具有将数据库从错误状态恢复到某一已知正确状态的功能，这就是数据库恢复功能。

综上所述，数据库是长期存储在计算机内有组织的大量的共享的数据集合。它可以供各种用户共享，具有最小冗余度和较高的数据独立性。DBMS 在数据库建立、运用和维护时对数据库进行统一控制，以保证数据的完整性、安全性，并在多用户同时使用数据库时进行并发控制，在发生故障后对系统进行恢复。

数据库系统的出现使信息系统以加工数据的程序为中心转向围绕共享的数据库为中心的新阶段。这样既便于数据的集中管理，又有利于应用程序的研制和维护，提高了数据的利用率和相容性，提高了决策的可靠性。

目前，数据库已经成为现代信息系统的不可分离的重要组成部分。具有数百万甚至数十亿字节信息的数据库已经普遍存在于科学技术、工业、农业、商业、服务业和政府部门的信息系统。20 世纪 80 年代后，不仅在大型机上，在多数微机上也配置了 DBMS，使数据库技术得到更加广泛的应用和普及。

1.1.3 数据模型

建立数据库系统离不开数据模型，数据模型是对现实世界的抽象。在数据库技术中，数据模型用于描述数据库的结构与语义，用于表示实体及实体间的联系。目前，在数据库领域中最常用的数据模型有四种：

（1）层次模型（Hierarchical Model）；

（2）网状模型（Network Model）；

（3）关系模型（Relational Model）；

（4）面向对象模型（Object Oriented Model）。

其中层次模型和网状模型统称为非关系模型。

非关系模型的数据库系统在 20 世纪 70 年代至 80 年代初非常流行，在数据库系统产品中占据了主导地位，现在已逐渐被关系模型的数据库系统取代，但在美国等一些国家里，由于早期开发的应用系统都是基于层次数据库或网状数据库系统的，因此目前仍有不少层次数据库或网状数据库系统在继续使用。

20 世纪 80 年代以来，面向对象的方法和技术在计算机各个领域，包括程序设计语言、软件工程、信息系统设计、计算机硬件设计等各方面都产生了深远的影响，也促进了数据库中面向对象数据模型的研究和发展。

1．层次模型

层次模型是数据库系统中最早出现的数据模型，它采用树形结构来表示各类实体及实体间的联系。

层次模型的数据结构如下。

在数据库中定义满足下面两个条件的基本层次联系的集合为层次模型。

（1）有且只有一个结点没有双亲结点，这个结点称为根结点；

（2）根以外的其他结点有且只有一个双亲结点。

在层次模型中，每个结点表示一个记录类型，记录之间的联系用结点之间的连线表示，这种联系是父子之间的一对多的联系。

层次数据库系统的典型代表是 IBM 公司的 IMS 数据库管理系统，这是 1968 年 IBM 公司推出的第一个大型的商用数据库管理系统，曾经得到广泛的使用。

2．网状模型

网状数据库系统采用网状模型作为数据的组织方式，它采用网状结构来表示各类实体及实体间的联系。

网状模型的数据结构如下。

在数据库中定义满足下面两个条件的基本层次联系的集合为网状模型。

（1）允许一个以上的结点无双亲；

（2）一个结点可以有多于一个的双亲。

网状数据模型的典型代表是 DBTG 系统，亦称 CODASYL 系统，是 20 世纪 70 年代数

据系统语言研究会 CODASYL 下属的数据库任务组提出的一个系统方案。DBTG 系统虽然不是实际的软件系统，但是它提出的基本概念、方法和技术对于网状数据库系统的研制和发展有重大的影响。后来不少的系统都采用 DBTG 模型或者简化的 DBTG 模型。例如，Cullinet Software 公司的 IDMS、Univac 公司的 DMS1100、Honeywell 公司的 IDS/2、HP 公司的 IMAGE 等。

3．关系模型

关系模型是目前最重要的一种数据模型。关系数据库系统采用关系模型作为数据的组织方式。

关系模型的数据结构如下。

在关系模型中，数据的逻辑结构就是一张二维表，它由行和列组成。在关系模型中，实体及实体间的联系都是用表来表示的。

1970 年美国 IBM 公司 San Jose 研究室的研究员 E.F.Codd 首次提出了数据库系统的关系模型，开创了数据库关系方法和关系数据理论的研究，为数据库技术奠定了理论基础。由于 E.F.Codd 的杰出工作，他于 1981 年获得 ACM 图灵奖。20 世纪 80 年代以来，计算机厂商新推出的数据库管理系统几乎都支持关系模型，非关系系统的产品也大都加上了关系接口。

4．面向对象模型

面向对象模型是用面向对象的观点来描述现实世界实体（对象）的逻辑组织、对象间限制、联系等的模型。面向对象的数据模型主要包含如下概念：

- 对象与对象标识
- 封装
- 类
- 类层次
- 消息

面向对象数据库是数据库技术与面向对象技术相结合的产物。人们认为面向对象数据库将成为下一代数据库系统的典型代表。在 20 世纪 80 年代后期，面向对象的概念得到了广泛的发展，经过反复的探索，大家已经对面向对象数据库的核心概念和面向对象数据库管理系统的基本目标取得了共识。但面向对象数据库系统在成为新一代数据库领军地位之前，还需要解决标准化和性能两个方面的问题。



1.2　关系型数据库基本概念

关系型数据库采用关系模型作为数据的组织方式，关系模型中有以下一些基本术语。

1.2.1　基本术语

- 关系（Relation）：一个关系对应一张表。
- 元组（Tuple）：表中的一行即为一个元组。
- 属性（Attribute）：表中的一列即为一个属性，为每一个属性起一个名字即为属性名。
- 主码（Prime Key）：也可称为主键，它是表中的某个属性组，可以唯一确定一个元组。
- 主属性（Prime Attribute）：主码所包含的所有属性。
- 外码（Foreign Key）：也可称为外键，如果关系 R 中包含另一个关系 S 的主属性 F，则称此主属性 F 为关系 R 的外码，并称关系 S 为参照关系，关系 R 为依赖关系。
- 域（Domain）：属性的取值范围。

尽管关系与传统的二维表格数据文件具有类似之处，但是它们又有区别，严格地说，关系是一种规范化的二维表格，具有如下性质：

- 属性值具有原子性，不可分解。
- 没有重复的元组。
- 理论上没有行序，但是有时使用时可以有行序。

1.2.2　Codd准则

关系数据库系统是目前应用最广泛的数据库系统。实际应用中各类关系产品的功能都是有差异的，根据其支持运算的不同，关系系统可分为（最小）关系系统、完备关系系统、全关系系统。1974 年 E.F.codd 提出了全关系系统的十二条基本准则，只有遵循这些准则的系统才是全关系系统。以下是具体的准则。

1．信息准则

关系数据库中的所有信息都应在逻辑一级上用一种方法（表中的值）显式地表示。

2．保证访问准则

依靠于表名、主键和列名，保证能以逻辑的方式访问数据库中的每个数据项。

3．空值的系统化处理

RDBMS 支持空值（不同于空的字符串或空白字符串，并且不为 0）系统化地表示缺少的信息，且与数据类型无关。

4．基于关系模型的联机目录

数据库的描述在逻辑上应该和普通数据采用同样的方式，使得授权用户可以使用查询一般数据所用的关系语言来查询数据库的描述信息。

5．统一的数据字语言准则

一个关系系统可以具有多种语言和多种终端使用方式（如表格填空方式、命令行方式等）。但是，必须有一种语言，它的语句可以表示为具有严格语法规定的字符串，并能全面地支持以下功能：数据定义、视图定义、数据操作（交互式或程序式）、完整约束、授权、事务控制（事务开始、提交、撤销）。

6．视图更新准则

所有理论上可更新的视图也应该允许由系统更新。

7．高阶的插入，更新和删除

把一个基本关系或导出关系作为一个操作对象进行数据的检索，以及插入、更新和删除。

8．数据的物理独立性

无论数据库的数据在存储表示上或存取方法上做何种变化，应用程序和终端活动都要保持逻辑上的不变性。

9．数据的逻辑独立性

当基本表进行理论上信息不受损害的任何变化时，应用程序和终端活动都要保持逻辑上的不变性。

10．数据完整性的独立性

关系数据库的完整性约束必须是用数据子语言定义并存储在目录中的，而不是在应用

程序中加以定义的。至少要支持以下两种约束：实体完整性，即主键中的属性不允许为NULL；参照完整性，即对于关系数据库中每个不同的非空的外码值，必须存在一个取自同一个域匹配的主键值。

11．分布的独立性

一个RDBMS应该具有分布独立性。分布独立性是指用户不必了解数据库是否是分布式的。

12．无破坏准则

如果 RDBMS 有一个低级语言（一次处理一个记录），这一低级语言不能违背或绕过完整性准则及高级关系语言（一次处理若干记录）表达的约束。

1.2.3　关系完整性

关系完整性规则是对关系的某种约束条件。关系模型中有三类完整性约束：实体完整性、参照完整性和用户定义完整性。其中实体完整性和参照完整性是关系模型必须满足的完整性约束条件，被称为关系的不变性，由关系系统自动支持。

- 实体完整性：关系的主属性不能为空值。空值（NULL）就是指不知道或是不能使用的值，注意它与数值 0 和空字符串的意义是不一样的。
- 参照完整性：如果关系 R1 的外码与关系 R2 的主码相符，那么关系 R1 的外码的每个值必须在关系 R2 的主键的值中找到或者是空值。
- 用户定义完整性：是针对某一具体的实际数据库的约束条件。它由应用环境所决定，反映某一具体应用所涉及的数据必须满足的要求。关系模型提供定义和检验这类完整性的机制，以便用统一的、系统的方法处理，而不必由应用程序承担这一功能。

1.3　SQL 语言概述

在关系数据库中数据存储在表中，而表是行和列的集合。根据关系数据理论和 Codd 准则的定义，一种语言必须能处理与数据库通信的所有问题，这种语言有时也称为“综合数据专用语言”。该语言在关系型数据库管理系统中就是结构化查询语言（SQL）。SQL 通过制定列、表以及它们之间的各种关系来检索或更新数据。它主要通过数据操作、数据定义和数据管理三种操作来实现，是在关系数据库中定义和处理数据的标准化语言。

1.3.1 SQL语言分类

SQL 的主要语句可分为以下三类。

- 数据控制语言（Data Control Language，DCL）：包括 GRANT 语句和 REVOKE 语句。
- 数据定义语言（Data Definition Language，DDL）：包括 CREATE 语句、DECLARE 语句、ALTER 语句和 DROP 语句。
- 数据操作语言（Data Manipulation Language，DML）：包括 SELECT 语句、INSERT 语句、UPDATE 语句和 DELETE 语句。

1.3.2 数据类型

值的数据类型决定了 DB2 如何解释该值。DB2 支持大量内置的数据类型，还支持用户定义的数据类型（User-Defined Data Types，UDT）。内置的数据类型可以分为数值型（numeric）、字符串型（character string）、图形字符串（graphic string）、二进制串型（binary string）或日期时间型（date time）。还有一种叫做 DATALINK 的特殊数据类型。DATALINK 值中包含了对存储在数据库以外的文件的引用。

1．数值型（numeric）

数值型数据类型包括 SMALLINT、INTEGER、BIGINT、DECIMAL、REAL 及 DOUBLE。所有的数字都有符号和精度，精度是指除符号以外的二进制位的个数或位数。如果数字的值大于或等于 0，就认为符号为正。

（1）小整型（SMALLINT）

小整型是两个字节的整数，精度为 5 位。小整型的范围是从–32 768～32 767。

（2）整型（INTEGER 或 INT）

整型是 4 个字节的整数，精度为 10 位。整型的范围是从–2 147 483 648～2 147 483 647。

（3）大整型（BIGINT）

大整型是 8 个字节的整数，精度为 19 位。大整型的范围是从–9 223 372 036 854 775 808～9 223 372 036 854 775 807。

（4）小数（DECIMAL(p,s)、DEC(p,s)、NUMERIC(p,s)或 NUM(p,s)）

小数类型的值是一种紧凑的十进制数，它有一个隐含的小数点。紧凑十进制数将以二-

十进制编码（Binary-Coded Decimal，BCD）来存储。小数点的位置取决于数字的精度（p）和小数位（s）。小数位是指数字的小数部分的位数，它不可以是负数，也不能大于精度。小数类型的最大精度是 31 位。十进制数的范围是从$-10^{31}+1$～$10^{31}-1$。

（5）单精度浮点数（Single-Precision Floating-Point，REAL）

单精度浮点数是实数的 32 位近似值。数字可以为 0，或者在从-3.402×10^{38}～1.175×10^{-37}或从1.175×10^{-37}～3.402×10^{38}的范围内。

（6）双精度浮点数（Double-Precision Floating-Point，DOUBLE、DOUBLE PRECISION 或 FLOAT）

双精度浮点数是实数的 64 位近似值。数字可以为 0，或者在从-1.79769×10^{308}～-2.225×10^{-307}或从2.225×10^{-307}～1.79769×10^{308}的范围内。

2．字符串型（character string）

字符串是字节序列。字符串包括 CHAR(*n*)类型的固定长度字符串和 VARCHAR(*n*)、LONG VARCHAR 或 CLOB(*n*)类型的可变长字符串。字符串的长度就是序列中的字节数。

（1）固定长度字符串（CHARACTER(*n*)）或 CHAR(*n*)）

固定长度字符串的长度介于 1～254 字节。如果没有指定长度，默认为 1 个字节。

（2）可变长度字符串（VARCHAR(*n*)、CHARACTER VARYING(*n*)或 CHAR VARYING(*n*)）

VARCHAR(*n*)类型的字符串是变长字符串，最长可达 32 672 字节。

（3）LONG VARCHAR

LONG VARCHAR 类型的字符串是变长字符串，最长可达 32 700 字节。

（4）字符大对象字符串（Character Large Object String，CLOB(*n*)）

字符大对象是长度可变的字符串，最长可达到 2 147 483 647 字节。只要指定了 *n*，那么 *n* 的值就是最大长度。

3．图形字符串（graphic string）

图形字符串是代表双字节字符数据的字节序列。图形字符串包括类型为 GRAPHIC(*n*)的定长图形字符串和类型为 VARGRAPHIC(*n*)、LONG VARGRAPHIC 和 DBCLOB(*n*)的变长图形字符串。字符串的长度就是序列中双字节字符的数目。

（1）定长图形字符串（Fixed-Length Graphic String，GRAPHIC(*n*)）

定长图形字符串的长度介于 1～127 个双字节字符。如果没有指定长度，就认为是 1 个

字节。

（2）变长图形字符串（Varying-Length Graphic String，VARGRAPHIC(*n*)）

VARGRAPHIC（*n*）类型的字符串是变长图形字符串，最大长度为 16 336 个双字节字符。

（3）LONG VARGRAPHIC

LONG VARGRAPHIC 类型的字符串是变长图形字符串，最大长度为 16 350 个双字节字符。

（4）双字节字符大对象字符串（Double-Byte Character Large Object String，DBCLOB(*n*)）

双字节字符大对象字符串是变长双字节字符图形字符串，最大长度为 1 073 741 823 个字符。如果指定 *n*，那么 *n* 就是最大长度。

4．二进制串型（binary string）

二进制串是字节序列。二进制串包括可变长度的 BLOB(*n*)类型的字符串，它用于容纳非传统型的数据，如图片、语音和混合媒体等，还可容纳用户定义类型及函数的结构化数据。二进制大对象是可变长的字符串，最长可达 2 147 483 647 字节。

5．日期时间型（date time）

日期时间数据类型包括 DATE、TIME 和 TIMESTAMP。日期时间可在某些算术和字符串操作中使用，而且兼容某些字符串，但它们既不是字符串也不是数字。

（1）DATE

DATE 是一个由三部分组成的值（年、月和日）。年份部分的范围从 0001～9999；月份部分的范围从 1～12；日部分的范围从 1～*n*，*n* 的值取决于月份。DATE 列长 10 个字节。

（2）TIME

TIME 是一个由三部分组成的值（小时、分和秒）。小时部分的范围从 0～24；分和秒部分的范围都是从 0～59。如果小时为 24，则分和秒的值都为 0。TIME 列长 8 个字节。

（3）TIMESTAMP

TIMESTAMP 是一个由 7 部分组成的值（年、月、日、小时、分、秒和微秒）。前 6 个部分的范围同上，微秒部分的范围从 000000～999999。TIMESTAMP 列的长度是 26 个字节。

6．专用寄存器

专用寄存器是数据库管理器为应用程序进程定义的存储区域，用于存储可以在 SQL 语句中引用的信息。目前，DB2 支持如下专用寄存器：

- CURRENT DATE

- CURRENT REFRESH AGE
- CURRENT DEFAULT TRANSFORM GROUP
- CURRENT SCHEMA
- CURRENT DEGREE CURRENT SERVER
- CURRENT EXPLAIN MODE
- CURRENT TIME
- CURRENT EXPLAIN SNAPSHOT
- CURRENT TIMESTAMP
- CURRENT NODE
- CURRENT TIMEZONE
- CURRENT PATH USER
- CURRENT QUERY OPTIMIZATION

1.3.3　数据控制语言（DCL）

1．GRANT语句

使用 GRANT 语句可以向单个用户或组显式授予权限和特权，授权对象包括数据库、表空间、表、视图、索引、包和模式。

GRANT 的语法如下：

```
GRANT privilege ON object-type object-name
TO {USER|GROUP|PUBLIC} authorization-name
WITH GRANT OPTION
```

PUBLIC 关键字表示将特权授予所有用户。

WITH GRANT OPTION 允许被授权的对象进一步将权限授予其他用户。

2．REVOKE语句

使用 REVOKE 语句可以显式撤销单个用户或组的权限和特权，撤销权限的对象包括数据库、表空间、表、视图、索引、包和模式。

REVOKE 的语法如下：

```
REVOKE privilege ON object-type object-name
FROM {USER|GROUP|PUBLIC} authorization-name
```

PUBLIC 关键字表示将撤销所有用户的特权。

1.3.4 数据定义语言（DDL）

1．CREATE语句

CREATE 语句用于创建数据库对象，包括：

- 数据库（DATABASE）
- 表（Table）
- 表空间（Table Space）
- 触发器（Trigger）
- 视图（View）
- 别名（Alias）
- 缓冲池（Buffer Pool）
- 函数（Function）
- 索引（Index）

……

每当用户创建数据库对象时，都会更新系统目录。如下所示是一个 CREATE 语句的示例，建立一张表：

```
CREATE TABLE org(
  depnum SMALLINT NOT NULL,
  depname VARCHAR(14),
  manager SMALLINT,
  division VARCHAR(10),
  location VARCHAR(13))
```

这条语句创建了一个 5 列的表，表名为 org。粗体小写字部分表示每个列的列名，非粗体大写字部分表示对应的数据类型。

2．DECLARE语句

DECLARE 语句和 CREATE 语句是一样的，只有一点例外，用它所创建的是只能在数据库连接期间存在的临时表。当要用到中间结果时，临时表是十分有用的。同任何其他表一样，声明过的表可以被引用，也可以被修改或删除。表是唯一可以被声明的对象。当用户声明临时表时，系统目录不会被更新。可以使用 DECLARE GLOBAL TEMPORARY

TABLE 语句来声明临时表，如：

```
DECLARE GLOBAL TEMPORARY TABLE session.temp1
  LIKE employee
  ON COMMIT PRESERVE ROWS
  NOT LOGGED
IN mytempspace
```

在这个示例中，DECLARE GLOBAL TEMPORARY TABLE 语句用来声明一个临时表，表名为 temp1，位于用户现有的一个名为 mytempspace 的临时表空间中。这张表的列名和定义同 employee 一样。每当处理 COMMIT 语句时，临时表中的行就会被保留下来（不会被删除）。最后，对临时表所做的更改不会记入日志。

3．ALTER语句

ALTER 语句可以用来改变现有数据库对象的一些特性，包括：

- 数据库（DATABASE）
- 表（Table）
- 表空间（Table Space）
- 触发器（Trigger）
- 视图（View）
- 别名（Alias）
- 缓冲池（Buffer Pool）
- 函数（Function）
- 索引（Index）

……

注意：不可以修改索引，如想要修改索引，就必须删除它然后用不同的定义创建新的索引。

4．DROP语句

DROP 语句可删除任何 CREATE 或 DECLARE 语句所创建的对象。DROP 语句将在删除数据库对象的同时也删除系统目录中关于该对象的定义。由于数据库对象之间可能存在某些依赖关系，所以删除对象可能会使有关的对象变成无效的状态。

1.3.5 数据操作语言（DML）

1．SELECT语句

SELECT 语句用于检索表或视图数据。

- 检索全表所有数据，如：SELECT * FROM table；
- FETCH FIRST，用于限制结果集显示的行数。
 如：SELECT * FROM table　FETCH FIRST 10 ROWS ONLY；
- 通过指定选择列表并用逗号分隔列名，可检索特定的列。
 如：SELECT col1，col3 FROM table；
- 使用 DISTINCT 子句来排除结果集中重复的行。
 如：SELECT DISTINCT col1 FROM table；
- 使用 AS 子句为选择列表上的表达式或项指定一个有意义的名字。
 如：SELECT col1，col2+col3 AS sum FROM table；
- WITH UR，用于指定使用只读方式查询数据。
 如：SELECT * FROM table WITH UR；

2．INSERT语句

INSERT 语句可以用于向表或视图中添加新行。向视图中插入一行，那么这一行也会被插入到该视图所基于的表中。可以使用 VALUES 子句来指定一行或多行的列数据。

例如：

```
INSERT INTO table VALUES(1, 2, 3);
```

该语句表明向表 table 中插入一条记录，三列的值分别为 1，2，3。

```
INSERT INTO table(col1, col3) VALUES(1, 3);
```

该语句表明向表 table 中插入一条记录，其中第一列和第三列的值分别为 1，3，第二列的值为空值（注意表定义中是否允许第二列值为空）。

3．UPDATE语句

UPDATE 语句用于改变表或视图中的数据。用户可以改变满足 WHERE 子句指定的条件的每一行中的一列或多列的值。如果不指定 WHERE 子句，则 DB2 会更新表或视图中的每一行。

例如：

```
UPDATE table SET col1=999;
```

该语句表明将全表的 col1 列均更新为 999。

```
UPDATE table SET col1=999 WHERE col1=1;
```

该语句表明将表中原 col1 列值为 1 的行均更新为 999。

```
UPDATE table SET col1=col1+100;
```

该语句表明将表中 col1 列的值均加上 100。

4．DELETE语句

DELETE 语句用于从表中删除整行数据。用户可以删除满足 WHERE 子句指定的条件的每一行。若不指定 WHERE 子句，那么 DB2 将删除表或视图中的所有行。

例如：

```
DELETE FROM table;
```

该语句表明删除全表的所有记录；

```
DELETE FROM table WHERE col1<>1;
```

该语句表明删除表中 col1 列值不为 1 的所有记录。

5．WHERE子句

通过 WHERE 子句可以指定若干选择条件或搜索条件，用来在表或视图中选择某些特定行。搜索条件由一个或多个谓词组成。谓词指定了行的某种条件，它可能是 True 或 False。建立搜索条件时需注意：

- 只对数值型数据类型应用算术运算；
- 只在兼容数据类型间进行比较；
- 字符型的值需用单引号；
- %和_是 DB2 中的字符通配符，%可代表多个字符，_代表 1 个字符。

6．ORDER BY子句

使用 ORDER BY 子句根据一列或多列中的值对结果集进行排序。ORDER BY 子句中指定的列名不一定要在选择的列表中指定。可以在 ORDER BY 子句中指定 DESC 来以降序

排列结果集，如果未写明则默认为 ASC，升序排列。

7．连接（Join）

连接是指把两个或两个以上的表中数据组合在一起进行查询。连接生成的结果集包含了多个表的列。为了方便后面的说明，我们定义如下两张表作为例子。

Table1：

ROW	REMARKS	VALUES
ROW1	Table1	1
ROW2	Table1	2
ROW3	Table1	3
ROW6	Table1	6

Table2：

ROW	REMARKS	VALUES
ROW1	Table2	1
ROW2	Table2	2
ROW3	Table2	3
ROW4	Table2	4
ROW5	Table2	5

（1）全连接（又称笛卡儿积）

全连接是最基本的连接，但并不实用。下面是一个全连接的例子：

```
SELECT * FROM Table1, Table2;
```

下面是连接后的结果片段：

ROW	REMARKS	VALUES	ROW	REMARKS	VALUES
ROW1	Table1	1	ROW1	Table2	1
ROW1	Table1	1	ROW2	Table2	2
ROW1	Table1	1	ROW3	Table2	3
ROW1	Table1	1	ROW4	Table2	4
ROW1	Table1	1	ROW5	Table2	5
ROW2	Table1	2	ROW1	Table2	1
ROW2	Table1	2	ROW2	Table2	2
……	……	……	……	……	……

（2）等值连接

等值连接是最常用的一种连接方式，可连接两张或多张表。

```
SELECT * FROM Table1 AS a, Table2 AS b WHERE a.values=b.values;
```

查询结果如下：

ROW	REMARKS	VALUES	ROW	REMARKS	VALUES
ROW1	TABLE1	1	ROW1	TABLE2	1
ROW2	TABLE1	2	ROW2	TABLE2	2
ROW3	TABLE1	3	ROW3	TABLE2	3

（3）不等值连接

等值连接是在 WHERE 子句中使用等号，而不等值连接则是在 WHERE 子句中使用了除等号以外的其他比较运算符。

```
SELECT * FROM Table1 AS a, Table2 AS b WHERE a.values<b.values;
```

查询结果片段如下：

ROW	REMARKS	VALUES	ROW	REMARKS	VALUES
ROW1	TABLE1	1	ROW2	TABLE2	2
ROW1	TABLE1	1	ROW3	TABLE2	3
ROW1	TABLE1	1	ROW4	TABLE2	4
ROW1	TABLE1	1	ROW5	TABLE2	5
ROW2	TABLE1	2	ROW3	TABLE2	3
ROW2	TABLE1	2	ROW4	TABLE2	4
ROW2	TABLE1	2	ROW5	TABLE2	5
ROW3	TABLE1	3	ROW4	TABLE2	4
ROW3	TABLE1	3	ROW5	TABLE2	5

（4）内连接（Inner Join）

内连接是只从笛卡儿积中返回满足连接条件的行。如果某一行在一个表中存在，但不在另一张表中，那么结果集中将不包括这一行。为了明确地说明内连接，可以通过在 FROM 子句中用 INNER JOIN 操作符及关键字 ON 为将要连接的表指定连接条件。

```
SELECT * FROM Table1 AS a JOIN Table2 AS b ON a.values=2;
```

查询结果如下：

ROW	REMARKS	VALUES	ROW	REMARKS	VALUES
ROW2	TABLE1	2	ROW1	TABLE2	1
ROW2	TABLE1	2	ROW2	TABLE2	2
ROW2	TABLE1	2	ROW3	TABLE2	3
ROW2	TABLE1	2	ROW4	TABLE2	4
ROW2	TABLE1	2	ROW5	TABLE2	5

（5）外连接（Outer Join）

外连接返回的是内连接操作生成的行，以及内连接操作无法返回的行。外连接共有如下两类。

① 左外连接

包括内连接和在左表中但内连接不会返回的那些行。这类连接在 FROM 子句中使用 LEFT OUTER JOIN 操作符。

```
SELECT * FROM Table1 a LEFT OUTER JOIN Table2 b ON a.values=2;
```

查询结果如下：

ROW	REMARKS	VALUES	ROW	REMARKS	VALUES
ROW1	TABLE1	1	----	-------	------
ROW2	TABLE1	2	ROW1	TABLE2	1
ROW2	TABLE1	2	ROW2	TABLE2	2
ROW2	TABLE1	2	ROW3	TABLE2	3
ROW2	TABLE1	2	ROW4	TABLE2	4
ROW2	TABLE1	2	ROW5	TABLE2	5
ROW3	TABLE1	3	----	-------	------
ROW6	TABLE1	6	----	-------	------

② 右外连接

包括内连接和在右表中但内连接不会返回的那些行。这类连接在 FROM 子句中使用 RIGHT OUTER JOIN 操作符。

```
SELECT * FROM Table1 a RIGHT OUTER JOIN Table2 b ON b.values=2;
```

查询结果如下：

ROW	REMARKS	VALUES	ROW	REMARKS	VALUES
----	-------	------	ROW1	TABLE2	1

续表

ROW	REMARKS	VALUES	ROW	REMARKS	VALUES
ROW1	TABLE1	1	ROW2	TABLE2	2
ROW2	TABLE1	2	ROW2	TABLE2	2
ROW3	TABLE1	3	ROW2	TABLE2	2
ROW6	TABLE1	6	ROW2	TABLE2	2
----	-------	------	ROW3	TABLE2	3
----	-------	------	ROW4	TABLE2	4
----	-------	------	ROW5	TABLE2	5

（6）表的自连接

表的自连接可以看成是两个相同表之间的直接连接，表的自连接是检查表中数据一致性的很好的办法。举个例子来看，就很好理解了。

假设有一张部门编号的表 org，如下：

DEPNUM	DEPNAME
1	办公室
2	技管办
3	安全部
4	系统部
5	测试中心
6	保卫部
4	***产品中心***

一个部门编号代表一个部门，且部门编号必须唯一。如果输入人员不小心输入了一条不符合要求的记录，见粗斜字部分，那么采用自连接方式就可以检查出这条不符合要求的记录。

```
SELECT *
FROM org A, org B
WHERE A.DEPNUM = B.DEPNUM
AND A.DEPNAME <> B.DEPNAME;
```

查询结果如下：

DEPNUM	DEPNAME	DEPNUM	DEPNAME
4	系统部	4	产品中心
4	产品中心	4	系统部

当然，如果在 DEPNUM 字段上建立 UNIQUE 的 INDEX，则可以在输入人员输入错误的记录之前发现这个问题。

8．集合运算

可以使用 UNION、UNION ALL 把两个或两个以上的查询合并成一个查询。

UNION 集合运算符会把两个或两个以上其他结果集合并生成一个结果集，并且将重复值删除掉；UNION ALL 与 UNION 类似，区别是它不删除重复值。

9．GROUP BY子句

在结果集中可以使用 GROUP BY 子句来组织行，每一组用结果集中的一行来表示。

10．子查询

子查询是把一个查询的返回结果作为参数提供给另一个查询的一种查询。如：

```
SELECT * FROM Table1 WHERE values IN
(SELECT values FROM Table2 WHERE row = 'row2');
```

查询结果如下：

ROW	REMARKS	VALUES
ROW2	TABLE1	2

说到子查询，还有一个需要特别说明的就是相关子查询，

如：

```
SELECT * FROM Table1 A WHERE A.values =
(SELECT B.values FROM Table2 B WHERE A.values=B.values);
```

查询结果如下：

ROW	REMARKS	VALUES
ROW1	TABLE1	1
ROW2	TABLE1	2
ROW3	TABLE1	3

通常一个相关子查询都可以转换为与其等价的连接。如上面的相关子查询如果转换成连接，则可以写成如下 SQL：

```
SELECT A.row,A.remarks,A.values FROM Table1 A,Table2 B
WHERE A.values = B.values;
```

11．HAVING和WHERE的区别

从功能角度看，HAVING 和 WHERE 是类似的，但是它们是对不同类型数据进行操作的。任何一条 SQL 语句都可以使用 WHERE 子句来指定要返回的行所需满足的条件。WHERE 子句对表、视图、同义词和别名中的数据进行操作。而 HAVING 子句则不同，它对分组的信息进行操作。只有使用了 GROUP BY 子句的 SQL 语句才能使用 HAVING 子句。

12．CASE表达式

CASE 语句根据指定的表达式的值，从多条语句中选择一条来执行。CASE 语句常用来替代对多个表进行组合的 UNION 语句。

如：

```
SELECT creator, name, 'TABLE'
   FROM SYSIBM.SYSTABLES
   WHERE type='T'
   UNION ALL
SELECT creator, name, 'VIEW'
   FROM SYSIBM.SYSTABLES
   WHERE type='V'
UNION ALL
SELECT creator, name, 'ALIAS'
   FROM SYSIBM.SYSTABLES
      WHERE type='A';
```

以上 SQL 可以用 CASE 语句简单编写成如下形式：

```
SELECT creator, name,
   CASE type
   WHEN 'T' THEN 'TABLE'
   WHEN 'V' THEN 'VIEW'
   WHEN 'A' THEN 'ALIAS'
        ELSE 'OTHER'
   END
FROM SYSIBM.SYSTABLES ;
```

使用 CASE 语句代替多个合并操作时，性能将可能有所提高，因为 DB2 在产生结果集时，对数据的访问次数更少。如上例，CASE 表达式只需要 1 遍而 UNION 需要 3 遍。

CASE 表达式的另一个有价值的用法是用来进行表的旋转。一个常见的需求就是对一个符合范式的表达式产生不符合范式的查询结果。

例如，假设有一张记录销售经理销售量的表 sales：

SALES_MGR	MONTH	YEAR	SALES_COUNT
JONE	1	2009	100
JONE	2	2009	150
……	……	……	……

使用 CASE 语句可以在一行中显示某个销售经理在某年中的 1～12 月各个月份的销售情况，这就是表的旋转。CASE 语句如下：

```
SELECT sales_mrg,
(CASE month WHEN 1 THEN sales_count ELSE null END） AS jan,
(CASE month WHEN 2 THEN sales_count ELSE null END) AS feb,
(CASE month WHEN 3 THEN sales_count ELSE null END) AS mar,
(CASE month WHEN 4 THEN sales_count ELSE null END) AS apr,
(CASE month WHEN 5 THEN sales_count ELSE null END) AS may,
(CASE month WHEN 6 THEN sales_count ELSE null END) AS jun,
(CASE month WHEN 7 THEN sales_count ELSE null END) AS jul,
(CASE month WHEN 8 THEN sales_count ELSE null END) AS aug,
(CASE month WHEN 9 THEN sales_count ELSE null END) AS sep,
(CASE month WHEN 10THEN sales_count ELSE null END) AS oct,
(CASE month WHEN 11 THEN sales_count ELSE null END) AS nov,
(CASE month WHEN 12 THEN sales_count ELSE null END) AS dec
FROM sales
WHERE sales_mgr='JOHN'
AND year=2009;
```

1.3.6 SQL函数

使用 SQL 语句对 DB2 表中的数据进行操作时，有两类内置的函数可供使用，它们是列函数和标量函数。可以使用这些函数进一步简化对复杂数据的访问。DB2 同时也提供了让用户创建自己定义函数的功能，称为“用户定义函数”。

1．列函数

列函数（Column Function）可从一组数据中计算出一个特定的列值或表达式值。它提供了汇集数据的能力，使用户能够在一条 SQL 语句中进行跨越多个行的数据统计和计算。使用列函数需注意如下几条规则：

- 列函数只能用在 SELECT 语句中。

- 对列函数不许显式地指定列名或表达式。
- 每个列函数对所做的 SELECT 操作的数据行集合只返回一个值。
- 如果对 SELECT 语句的某个列使用列函数，除非也使用 GROUP BY 子句，否则必须对同一个 SELECT 语句中的其他所有列也使用这个列函数。
- 使用 GROUP BY 子句来对一组命令列使用列函数，任何在这条 SELECT 语句中的命令列也必将被这个列函数处理。
- 除 COUNT 和 COUNT_BIG 函数外，列函数的结果值与它处理的列具有相同的数据类型。COUNT 列函数返回整型值，而 COUNT_BIG 列函数返回小数值。
- 如果预先在 WHERE 子句中定义的条件没有返回数据，而是返回空值 NULL，则列函数将不返回，SQLCODE 为 100。
- 当在可以取空值的列上使用 AVG、MAX、MIN、STDDEV、SUM 和 VARIANCE 函数时，应在使用这些函数之前将所有的空值剔除。
- 在使用某个列函数之前，可以使用 DISTINCT 关键字来去除重复值，DISTINCT 对 MAX 和 MIN 函数无效。
- 可以使用 ALL 关键字指出重复的值不被去除，ALL 在列函数使用中是默认的。
- 只有当一个 WHERE 子句是 HAVING 子句的子查询的一部分时，列函数才能在 WHERE 子句中被定义。
- 在列函数表达式中指定的每个列名都必须被相同的组所引用。

常用的列函数有以下几类。

（1）AVG 函数

AVG 函数计算列或表达式中的值的平均值，它只能对数字型的参数进行运算操作。注意：AVG(Col)和 SUM(COL)/COUNT(*)可能返回不同的值，原因在于 COUNT 函数将计算所有的列而不管其值是否为空，而 SUM 函数将忽略空值。

（2）COUNT 函数

COUNT 函数是对某个表的行数进行计数，或者对某个给定的具有不同值的列进行计数。COUNT 函数可以在列一级或行一级上进行操作，相应的语法有所不同。COUNT 函数只是简单地对行数进行计数并返回结果值，并不关心所计数的行中存储的数据值。

（3）COUNT_BIG 函数

COUNT_BIG 函数和 COUNT 函数类似，它对一个数据库表中的行数或某个列的不同值的个数进行计数。不同之处在于 COUNT_BIG 函数返回 DECIMAL(31，0)数据类型的结果值，而 COUNT 函数返回的结果是整数值，其最大值为 DB2 可以表示的最大整数值。

除了返回值是小数类型外，COUNT_BIG 与 COUNT 完成同样的工作，所以使用 COUNT 函数的 SQL 语句可以简单地用 COUNT_BIG 函数来替换。COUNT_BIG 函数和 COUNT 函数有相同的限制，它的参数是除去大对象类型（CLOB、DBCLOB 或 BLOB）外任何系统内置的数据类型。字符串类型的参数不能长于 255 字节，而图形串类型的参数不能长于 127 字节。

（4）MAX 函数

MAX 函数返回指定的列或表达式值中的最大值。

MAX 函数返回结果应当与指定的列或表达式具有相同的数据类型。它的参数是除去大对象类型（CLOB、DBCLOB 或 BLOB）外任何系统内置的数据类型。字符串类型的参数不能长于 255 字节，而图形串类型的参数不能长于 127 字节。

（5）MIN 函数

MIN 函数返回指定的列或表达式值中的最小值。

MIN 函数返回结果应当与指定的列或表达式具有相同的数据类型。它的参数是除去大对象类型（CLOB、DBCLOB 或 BLOB）外任何系统内置的数据类型。字符串类型的参数不能长于 255 字节，而图形串类型的参数不能长于 127 字节。

（6）STDDEV 函数

STDDEV 函数返回一组数的标准均方差，它的参数可以是系统定义的任何数值型数据类型，返回的结果是双精度浮点数。

（7）SUM 函数

SUM 函数返回对指定列或表达式的值的累加和，它的参数可以使用任何系统定义的数值型数据，返回值必须在其数据类型可以允许接受的值的范围之内。

除下列情况之外，函数返回值的数据类型必须同参数值的数据类型相同：

- 对 SMALLINT 值求和将返回 INTEGER 值；
- 对单精度数求和将返回双精度值。

（8）VARIANCE 函数

VARIANCE 函数返回一组数据的方差，它的参数可以使用任何系统定义的数值型数据，返回值为双精度浮点数，VARIANCE 可简写为 VAR。

2．标量函数

标量函数（Scalar Function）是应用在某个列或表达式上对单个值进行运算处理的函数，而列函数是应用在某个数据集合上的。标量函数将某个列或表达式的值进行转换并将转换结果作为返回值。DB2 提供的标量函数如下。

- ABS：返回数的绝对值。
- HEX：返回值的十六进制表示。
- LENGTH：返回自变量中的字节数（对于图形字符串则返回双字节字符数）。
- YEAR：抽取日期时间值的年份部分。

1.3.7　制定约束

约束是数据库管理程序实施的规则，DB2 包含 4 种类型的约束处理。

- 唯一性约束：确保表中的关键字值是唯一的。检查对组成主关键字的列的任何更改，以保证唯一性。
- 参照完整性约束：在插入、更新和删除操作上实现参考约束。所有外部关键字的所有值都有效才是数据库的一个正确状态。
- 表检查约束：验证更改后的数据有无违反创建或更改表时指定的条件。
- 触发器：定义要执行的一组操作，当对指定的表执行更新、删除或插入操作时要调用这组操作。

1．唯一性约束

唯一性约束是一个规则，它确保关键字值在表中是唯一的。在唯一约束中组成该关键字的每一列必须定义为 NOT NULL。可在使用 PRIMARY KEY 子句或 UNIQUE 子句的 CREATE TABLE 或 ALTER TABLE 语句中定义唯一约束。

一个表可有任意多个唯一约束，但是只能定义一个唯一约束作为一个表的主键。此外，一个表在相同的一组列上不能有多个唯一约束。当定义一个唯一约束时，DB2 就会创建一个唯一索引并将它指定为系统必需的主索引或唯一索引。此约束通过唯一索引来实现。一旦在某列上建立了唯一约束，则在多行更新期间对唯一性的检查将延迟到更新结束后进行。

2．参照完整性约束

DB2 通过参照约束维护参照完整性。参照完整性要求子表给定的属性的所有值存在于父表的某些列中。参照约束是指一个指定的外部关键字的非 NULL 值只有是指定表的唯一关键字的值时才有效。参照约束的目的是保证数据库表之间的关系得到维持，并遵守数据输入规则。

对于参照约束的表，在 INSERT、DELETE、UPDATE 和 DROP 等 SQL 操作上有某些限制。

（1）INSERT 规则

可以随时在父表中插入一行，而不必在子表中执行任何操作。但不能在子表中插入行，除非在父表中有一行键值等于要插入的行的外键，或该外键值是 NULL。

（2）DELETE 规则

当从父表中删除一行时，需检查在子表中是否有某行或某些行的外键值等于父表被删除的行的键值。若不存在则可正确删除；若存在，是否可以删除该行由创建子表时指定的删除规则决定。

① 若定义为 RESTRICT，则阻止父表删除任何行，如果一定要删除，需先删除子表对应的行再删除父表的行才可实现。

② 若定义为 NOACTION，则强制要求父表不能删除行。

③ 若定义为 CASCADE，则在删除父表的行后自动删除子表对应的行，无须首先删除子表对应的行。若子表还有关联的子表，则再对之后的子表实施删除规则，DB2 可实现级联删除。

④ 若定义为 SET NULL，在删除父表的行后将子表中的外键值置为 NULL，而该行的其他部分保持不变。

⑤ 若创建子表时未定义删除规则，则默认为 NOACTION 规则。

在一个删除操作中可能涉及的任何表称为删除连接的表。下列限制应用于删除连接的关系：

① 当多个表组成的一个参照循环中，一个表不能与它自己是删除连接关系。

② 当一个表通过多个从属关系与另一个表形成删除连接关系时，这些关系必须有相同的删除规则 CASCADE 或 NOACTION。

③ 当一个自参照表是 CASCADE 关系中另一个表的子表时，自参照关系的删除规则也必须是 CASCADE。

（3）UPDATE 规则

若需要更新子表的外键，而该外键值不为 NULL 时，它必须与父表的某个键值匹配，否则不允许更新任何行。

若更新父表的某个键值时：

① 若子表中存在某些行与父表的该键值匹配，且更新规则是 RESTRICT 时拒绝该更新。

② 若更新语句完成时（触发后的情况除外）子表中的任何行没有对应的父表键值，当更新规则为 NOACTION 时拒绝该更新。

要更新父表的该行，必须通过下列操作首先除去与子表的某些子行的关系：

- 删除子行。
- 更新子表中的外键值，以包括另一个有效的关键字。

3．表检查约束

表检查约束指定应用于一个表的每一行的搜索条件。当对表进行更新或插入操作时，自动激活这些约束。表检查约束是通过 CREATE TABLE 或 ALTER TABLE 语句定义的。

4．触发器

触发器是每当对指定的表执行删除、插入或更新操作时执行的一组定义的操作。因为触发器存储在数据库中，所以不必在每个应用程序中编写操作码。触发器只编码一次并存储在数据库中，当应用程序使用该数据库时，会在需要时由 DB2 自动调用触发器，确保始终实施与该数据相关的规则。若规则需要修改，只需修改触发器即可。

1.3.8　静态SQL与动态SQL

静态 SQL 指的是在应用程序中直接嵌入的 SQL 语句，这些 SQL 语句在运行的时候不能被更新。另外，静态 SQL 需要在程序运行之前进行预编译处理。DB2 预编译器将对 SQL 语句进行语法检查，并将其转换为宿主语言及生成宿主语言语句以调用 DB2。

静态 SQL 的优点：

- 提高运行时的速度；
- 经过了编译错误检查。

静态 SQL 的缺点：

- 灵活性差；
- 需要更多的代码，因为查询条件不能在运行时进行变更。

动态 SQL 像静态 SQL 一样，同样需要进行预编译，但不同的是，动态 SQL 语句是在程序运行时构建 SQL 语句并把这些语句提交给数据库引擎，之后再将数据返回给程序。



1.4 关系数据库设计

1.4.1 关系规范化

在数据库中，数据之间存在着密切的联系。关系数据库由相互联系的一组关系组成，每个关系包括关系模式和关系值两个方面。关系模式是对关系的抽象定义，给出关系的具体结构；关系的值是关系的具体内容，反映关系在某一时刻的状态。一个关系包含许多元组，每个元组都是符合关系模式结构的一个具体值，并且都分属于相应的属性。在关系数据库中，每个关系都需要进行规范化，使之达到一定的规范化程度，从而提高数据的结构化、共享性、一致性和可操作性。

关系模型原理的核心内容就是规范化概念，规范化就是把数据库组织成在保持存储数据完整性的同时最小化冗余数据的结构的过程。规范化的数据库必须符合关系模型的范式规则。范式可以防止在使用数据库时出现不一致的数据，并防止数据丢失。关系模型的范式有第一范式、第二范式、第三范式和 BCNF 范式等多种。

在这些定义中，高级范式根据定义属于所有低级的范式。第三范式中的关系属于第二范式，第二范式中的关系属于第一范式。下面将介绍规范化的过程。

1．第一范式（1NF）

第一范式是第二和第三范式的基础，是最基本的范式。第一范式包括以下原则：

- 元组的每个属性只可以包含一个值。
- 关系中的每个元组必须包含相同数量的值。
- 关系中的每个元组一定不能相同。

如果关系模式 R 中的所有属性值都是不可再分解的原子值，那么就称此关系 R 是属于第一范式（First Normal Form，简称 1NF）的关系模式。在关系型数据库管理系统中，涉及的研究对象都是满足 1NF 的规范化关系，不是 1NF 的关系称为非规范化的关系。

例如，如下是一个电脑配件经销商的关系，其中的第四和第五行的 2、3 数组违反了第一范式，因为“商品编号”和“商品名称”的属性都包含两个值。

库存编号	商品编号	商品名称	单　　价	库存数量	供应商名称
1	1001	主机	250	15	中达
2	1005	CD 光驱	240	10	海月
3	1006	CPU	960	10	海月
4	1002、1006	主机、CPU	800/960	20	海月
5	1003、1006	主机、CPU	450/960	15	畅通

如果要将这些数据规范化，就必须将包含多个属性值的属性进一步分解，使每个属性只包含一个值，每个数组包含相同数量的值，并且每个数组各不相同，如下所示，将之前的 1 个关系拆分成 3 个关系，这时的数据才符合第一范式。

库存编号	商品编号	库存数量
1	1001	15
2	1002	20
3	1003	15
4	1005	10
5	1006	10

商品编号	商品名称	单　　价
1001	主机	250
1002	主机	800
1003	主机	450
1005	CD 光驱	240
1006	CPU	960

供应商编号	供应商名称
1	中达
2	海月
3	畅通

2．第二范式（2NF）

第二范式（2NF）规定关系必须满足第一范式的要求，并且关系中的所有属性依赖于整个候选键。候选键是一个或多个唯一标识每个数据组的属性集合。

在如下关系中，可以将“商品名称”和“供应商名称”指定为候选键。这些值共同唯一标识每个元组。而在该关系中，“库存编号”属性只依赖于“商品名称”，而不依赖于“供应商名称”属性，所以该关系并不满足第二范式的要求，需要进一步拆分。

库存编号	商品编号	商品名称	单　　价	库存数量	供应商名称
1	1001	主机	250	15	中达
2	1002	主机	800	20	海月
3	1003	主机	450	15	畅通
4	1005	CD 光驱	240	10	海月
5	1006	CPU	960	10	海月

3．第三范式（3NF）

第三范式（3NF）同第二范式一样依赖于关系的候选键。为了遵循 3NF 的指导原则，关系必须在 2NF 中，非键属性相互之间必须无关，并且必须依赖于键。

在如下所示的关系中，候选键“供应商代号”是属性。“商品名称”和“供应商名称”的属性都依赖于主键“库存编号”，并且相互之间进行关联。“供应商代号”属性依赖于“商品编号”，而不依赖于主键“库存编号”。

库存编号	商品编号	商品名称	单　　价	库存数量	供应商代号	供应商名称
1	1001	主机	250	15	5	中达
2	1002	主机	800	20	4	海月
3	1003	主机	450	15	1	畅通
4	1005	CD 光驱	240	10	4	海月
5	1006	CPU	960	10	4	海月

在关系数据库中，对关系模式的基本要求是满足第一范式。这样的关系模式就是合法、允许的。但是，仅仅满足第一范式的关系模式在使用过程中存在插入、删除异常，修改复杂，数据冗余等毛病。而规范化的目的正是要解决这些问题。因此，规范化的基本思想就是逐步消除数据依赖中不合适的部分，使模式的各关系模式达到某种程度的“分离”，即“一事一地”的模式设计原则。让一个关系描述一个概念、一个实体或实体间的一种联系。若多于一个概念就把它“分离”出去。因此所谓规范化实质上是概念的单一化。

对于关系设计，理想的设计目标是按照规范化规则设计关系。但是，在现实世界中，各实体之间不可能仅存在一种关系，也就是说，并不是规范化程度越高，模式就越好。在实际的数据库设计过程中，逆规范化也是通用的惯例，也就是说需要违反规范化规则，尤

其是违反第二范式和第三范式。当过于规范化的结构使实现方式复杂化时，逆规范化主要用于提高性能或减少复杂性。

1.4.2　实体——关系模型

关系数据库的数据模型有概念模型，其典型代表就是 P.P.S.Chen 于 1976 年提出的“实体-关系模型”（E-R 模型），是用户和数据库设计人员之间进行交流的工具，在设计数据库系统之前，需要使用 E-R 图将现实世界中的实体和实体之间的联系转换为概念模型。E-R 模型的基本元素是：实体、属性和联系。E-R 图提供了表示实体、属性和联系的方法。

1．实体（Entity）

实体是一个数据对象，指可以区别客观存在的事物，如人、部门、表格、项目等。同一类实体的所有实例就构成该对象的实体集（entity classes）。也就是说，实体集是实体的集合，由该集合中实体的结构形式表示，而实例则是实体集中的某一个特例。

在 E-R 模型中，实体用方框表示，方框内注明实体的命名。实体名常用以大写字母开头的有具体意义的英文名词表示，联系名和属性名也采用这种方式。通常实体集中有多个实体实例。图 1.1 所示即为一个销售实体及两个销售实例。

SalesEntity	Sales 实例 1	Sales 实例 2
Stor_id	6380	7066
Ord_num	7222A	A2976
Ord_date	2009-05-10	2009-07-12
Qty	50	75
Payterms	Net 60	Net 30
Title_id	PS2091	TC3250

图 1.1　实体与实例的关系

2．属性（Attribute）

属性用于描述实体的特征。在上面关于实体的例子中，订单号、订单数量、订单日期等均属于订单该实体的属性。E-R 模型中假定实体集的所有实例具有相同的属性。同时，依据系统的需求，每个属性都有它的数据类型及特性。特性包括指定该属性在某些情况下是否必需、属性是否有默认值、属性的取值范围、是否为主键或候选键等。

3．码（Key）

唯一标识实体的属性集称为码。例如，学号是学生实体的码。

4．域（Domain）

属性的取值范围称为该属性的域。

5．实体集（Entity Set）

同型实体的集合称为实体集。例如，全体学生就是一个实体集。

6．联系（Relationship）

在现实世界中，事物内部及事物之间是有联系的，这些联系在信息世界中反映为实体内部的联系和实体之间的联系。实体内部的联系通常是指组成实体的各属性之间的联系，实体之间的联系通常是指不同实体集之间的联系。

通常实体之间的联系分为三类：

- 一对一联系（1∶1）
- 一对多联系（1∶*n*）
- 多对多联系（*m*∶*n*）

1.4.3 关系数据库设计的基本步骤

按照规范设计的方法，同时考虑到数据库开发的全过程，可以将数据库设计分为 6 个阶段：

- 需求分析阶段
- 概念结构设计阶段
- 逻辑结构设计阶段
- 物理设计阶段
- 数据库实施阶段
- 数据库运行和维护阶段

1.4.3.1 需求分析

需求分析的任务是通过详细调查现实世界要处理的对象（组织、部门、企业等），充分

了解原系统（手工系统或计算机系统）工作概况，明确用户的各种需求，然后在此基础上确定新系统的功能。新系统必须充分考虑今后可能的扩充和改变，不能仅仅按当前应用需求来设计数据库。

需求分析的重点是调查、收集与分析用户在数据管理中的信息要求、处理要求、安全性与完整性要求。

1. 信息要求

是指用户需要从数据库中获得信息的内容与性质。由用户的信息要求可以导出数据要求，即在数据库中需要存储哪些数据。

2. 处理要求

是指用户要求完成什么处理功能，对处理的响应时间有什么要求，处理方式是批处理还是联机处理。新系统的功能必须能够满足用户的信息要求、处理要求。

3. 安全性与完整性要求

确定用户的最终需求其实是一件很困难的事，这是因为一方面用户缺少计算机知识，开始时无法确定计算机究竟能为自己做什么，不能做什么，因此无法一下子准确地表达自己的需求，他们所提出的需求往往不断地变化。另一方面设计人员缺少用户的专业知识，不易理解用户的真正需求，甚至误解用户的需求。此外新的硬件、软件技术的出现也会使用户需求发生变化。因此设计人员必须与用户不断深入地进行交流，才能逐步确定用户的实际需求。

4. 需求分析的方法

调查用户需求的具体步骤如下。

STEP 01 调查组织机构情况，包括了解该组织的部门组成情况、各部门的职责等，为分析信息流程做准备。

STEP 02 调查各部门的业务活动情况，包括了解各部门输入和使用什么数据，如何加工处理这些数据，输出什么数据，输出格式如何，输出到什么部门。

STEP 03 协助用户明确对新系统的各种要求，包括信息要求、处理要求、完全性与完整性要求。

STEP 04 确定新系统的边界，对之前的调查结果进行初步分析，确定新系统应该实现的

功能。

常用的调查方法有：跟班作业、开调查会、请专人介绍、询问、问卷调查、查阅记录，等等。

1.4.3.2 概念结构设计

概念设计的重点在于信息结构的设计，它是整个数据库系统设计的关键。它独立于逻辑结构设计和 DBMS。描述概念模型的有力工具是 E-R 模型。设计概念结构通常有 4 类方法：

- 自顶向下，即首先定义全局概念结构的框架，然后逐步细化。
- 自底向上，即首先定义各个局部应用的概念结构，然后将它们集成起来形成全局概念结构。
- 逐步扩张，首先定义最重要的核心概念结构，然后向外扩充，以滚雪球的方式逐步生成其他概念结构，直至总体概念结构。
- 混合策略，将自顶向下和自底向上相结合，用自顶向下策略设计一个全局概念结构的框架，以它为骨架集成由自底向上策略中设计的各局部概念结构。

其中最经常使用的策略是自底向上的方法。它通常分为两步：

STEP 01 首先要根据需求分析的结果（数据流图、数据字典等）对现实世界的数据进行抽象，设计各个局部视图即分 E-R 图。标定局部应用中的实体、实体的属性、标识实体的码，确定实体之间的联系及其类型（1∶1、1∶n、m∶n）。

现实世界中一组具有某些共同特性和行为的对象就可以抽象为一个实体。对象和实体之间是“is MEMBER of”的关系。例如在学校环境中，可以把张三、李四、王五等对象抽象为学生实体。

对象类型的组成成分可以抽象为实体的属性。组成成分与对象类型之间是“is part of”的关系。例如学号、姓名、专业、年级等可以抽象为学生实体的属性。其中学号为标识学生实体的码。

实际上实体与属性是相对而言的，很难有截然划分的界限。同一事物，在一种应用环境中作为“属性”，在另一种应用环境中就必须作为“实体”。一般说来，在给定的应用环境中：

（1）属性不能再具有需要描述的性质。即属性必须是不可分的数据项。

（2）属性不能与其他实体具有联系。联系只发生在实体之间。

例如，学籍管理局部应用中主要涉及的实体包括学生、宿舍、档案材料、班级、班主任。那么，这些实体之间的联系又是怎样的呢？

由于一个宿舍可以住多个学生，而一个学生只能住在某一个宿舍中，因此宿舍与学生之间是 1∶n 的联系。由于一个班级往往有若干名学生，而一个学生只能属于一个班级，因此班级与学生之间也是 1∶n 的联系。由于班主任同时还要教课，因此班主任与学生之间存在指导联系，一个班主任要教多名学生，而一个学生只对应一个班主任，因此班主任与学生之间也是 1∶n 的联系。而学生和他自己的档案材料之间，班级与班主任之间都是 1∶1 的联系。

在一般情况下，性别通常作为学生实体的属性，但在本局部应用中，由于宿舍分配与学生性别有关，根据准则 2，应该把性别作为实体对待。

这样，得到学籍管理局部应用的分 E-R 图，如图 1.2 所示。

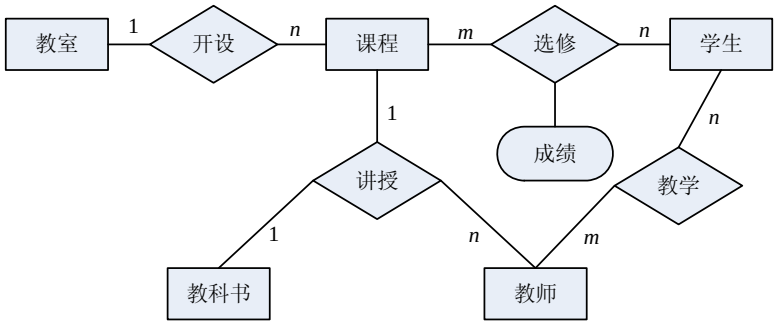


图 1.2　学籍管理局部应用的分 E-R 图

各实体的属性分别为：

学生：{姓名，学号，性别，年龄，所在系，年级，平均成绩}

课程：{课程号，课程名，学分}

教师：{职工号，姓名，性别，职称}

教科书：{书号，书名，价钱}

教室：{教室编号，地址，容量}

STEP 02 集成局部视图，即设计全局 E-R 图。

集成局部 E-R 图时需要两步：（1）合并；（2）修改与重构。

而各分 E-R 图之间的冲突主要有三类：属性冲突、命名冲突和结构冲突。

1. 属性冲突

（1）属性域冲突，即属性值的类型、取值范围或取值集合不同。

（2）属性取值单位冲突。

2. 命名冲突

（1）同名异义，即不同意义的对象在不同的局部应用中具有相同的名字。

（2）异名同义（一义多名），即同一意义的对象在不同的局部应用中具有不同的名字。

3. 结构冲突

（1）同一对象在不同应用中具有不同的抽象。例如“课程”在某一局部应用中被当做实体，而在另一局部应用中则被当做属性。

（2）同一实体在不同局部视图中所包含的属性不完全相同，或者属性的排列次序不完全相同。

实体之间的联系在不同局部视图中呈现不同的类型。例如实体 E1 与 E2 在局部应用 A 中是多对多联系，而在局部应用 B 中是一对多联系；又如在局部应用 X 中 E1 与 E2 发生联系，而在局部应用 Y 中 E1、E2、E3 三者之间有联系。

解决方法是根据应用的语义对实体联系的类型进行综合或调整。

下面我们来看看如何生成学校管理系统的初步 E-R 图。我们着重介绍学籍管理局部视图与课程管理局部视图的合并。这两个分 E-R 图存在着多方面的冲突：

班主任实际上也属于教师，也就是说学籍管理中的班主任实体与课程管理中的教师实体在一定程度上属于异名同义，可以应将学籍管理中的班主任实体与课程管理中的教师实体统一称为教师，统一后教师实体的属性构成为：

教师：{职工号，姓名，性别，职称，是否为班主任}

将班主任改为教师后，教师与学生之间的联系在两个局部视图中呈现两种不同的类型，一种是学籍管理中教师与学生之间的指导联系，一种是课程管理中教师与学生之间的教学联系，由于指导联系实际上可以包含在教学联系之中，因此可以将这两种联系综合为教学联系。

在两个局部 E-R 图中，学生实体属性组成及次序都存在差异，应将所有属性综合，并重新调整次序。假设调整结果为：

学生：{学号，姓名，出生日期，年龄，所在系，年级，平均成绩}

解决上述冲突后，学籍管理分 E-R 图与课程管理分 E-R 图合并为初步 E-R 图。之后需修改、重构初步 E-R 图以消除冗余。图 1.3 是进行修改和重构后生成的基本 E-R 图。

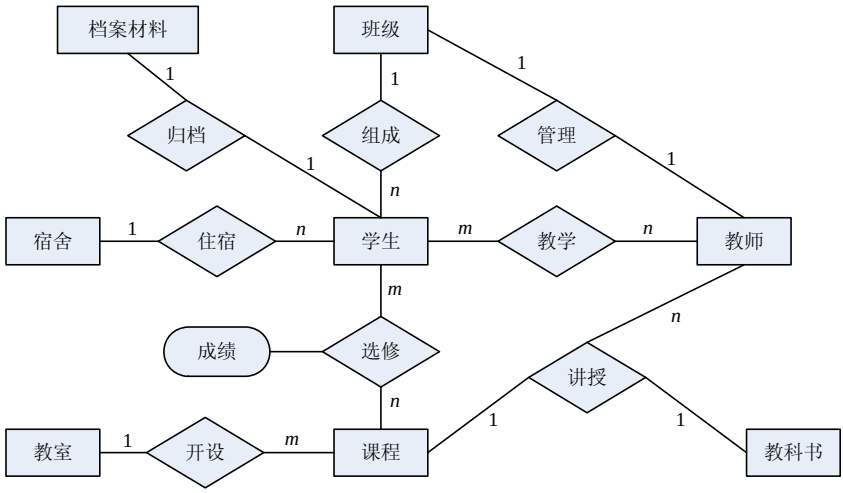


图 1.3　修改后的 E-R 图

1.4.3.3　逻辑结构设计

设计逻辑结构应该选择最适合描述与表达相应概念结构的数据模型，然后选择最合适的 DBMS。设计逻辑结构时一般要分三步进行：

STEP 01 将概念结构转换为一般的关系、网状、层次模型；

STEP 02 将转化来的关系、网状、层次模型向特定 DBMS 支持下的数据模型转换；

STEP 03 对数据模型进行优化。

1. 将 E-R 模型转换为关系模型

关系模型的逻辑结构是一组关系模式的集合。而 E-R 图则是由实体、实体的属性和实体之间的联系三个要素组成的。所以将 E-R 图转换为关系模型实际上就是要将实体、实体的属性和实体之间的联系转化为关系模式，这种转换一般遵循如下原则：

（1）一个实体型转换为一个关系模式。实体的属性就是关系的属性。实体的码就是关系的码。

（2）一个 m:n 联系转换为一个关系模式。与该联系相连的各实体的码及联系本身的属性均转换为关系的属性。而关系的码为各实体码的组合。

（3）一个 1:n 联系可以转换为一个独立的关系模式，也可以与 n 端对应的关系模式合并。如果转换为一个独立的关系模式，则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，而关系的码为 n 端实体的码。

（4）一个 1∶1 联系可以转换为一个独立的关系模式，也可以与任意一端对应的关系模式合并。如果转换为一个独立的关系模式，则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，每个实体的码均是该关系的候选码。如果与某一端对应的关系模式合并，则需要在该关系模式的属性中加入另一个关系模式的码和联系本身的属性。

（5）三个或三个以上实体间的一个多元联系转换为一个关系模式。与该多元联系相连的各实体的码以及联系本身的属性均转换为关系的属性。而关系的码为各实体码的组合。

（6）同一实体集的实体间的联系，即自联系，也可按上述 1∶1、1∶*n* 和 *m*∶*n* 三种情况分别处理。

（7）具有相同码的关系模式可合并。

2. 数据模型的优化

数据库逻辑设计的结果不是唯一的。为了进一步提高数据库应用系统的性能，通常以规范化理论为指导，还应该适当地修改、调整数据模型的结构，这就是数据模型的优化。

数据模型的优化方法为：

（1）确定数据依赖。

（2）对于各个关系模式之间的数据依赖进行极小化处理，消除冗余的联系。

（3）按照数据依赖的理论对关系模式逐一进行分析，考查是否存在部分函数依赖、传递函数依赖、多值依赖等，确定各关系模式分别属于第几范式。

（4）按照需求分析阶段得到的各种应用对数据处理的要求，分析对于这样的应用环境这些模式是否合适，确定是否要对它们进行合并或分解。

（5）对关系模式进行必要的分解。

规范化理论为数据库设计人员判断关系模式优劣提供了理论标准，可用来预测模式可能出现的问题，但是并不是规范化程序越高的关系就越优。

3. 设计外模式

前面我们根据用户需求设计了局部应用视图，这种局部应用视图只是概念模型，用 E-R 图表示。在我们将概念模型转换为逻辑模型后，即生成了整个应用系统的模式后，还应该根据局部应用需求，结合具体 DBMS 的特点，设计用户的外模式。

目前关系数据库管理系统一般都提供了视图概念，支持用户的虚拟视图。我们可以利用这一功能设计更符合局部用户需要的用户外模式。

定义数据库模式主要是从系统的时间效率、空间效率、易维护等角度出发的。由于用

户外模式与模式是独立的，因此我们在定义用户外模式时应该更注重考虑用户的习惯与方便。包括：

（1）使用更符合用户习惯的别名；

（2）针对不同级别的用户定义不同的外模式，以满足系统对安全性的要求；

（3）简化用户对系统的使用。

1.4.3.4　物理结构设计

数据库在物理设备上的存储结构与存取方法称为数据库的物理结构，为一个给定的逻辑数据模型选取一个最适合应用要求的物理结构的过程，就是数据库的物理设计。

数据库的物理设计通常分为两步：

STEP 01 确定数据库的物理结构；

STEP 02 对物理结构进行评价，评价的重点是时间和空间效率。

1. 确定数据库的物理结构

（1）确定数据的存储结构

确定数据库存储结构时要综合考虑存取时间、存储空间利用率和维护代价三方面的因素。这三个方面常常是相互矛盾的，例如消除一切冗余数据虽然能够节约存储空间，但往往会导致检索代价的增加，因此必须进行权衡，选择一个折中方案。聚簇功能不但适用于单个关系，也适用于多个关系。假设用户经常要按系别查询学生成绩单，这一查询涉及学生关系和课程关系的连接操作，即需要按学号连接。

（2）设计数据的存取路径

在关系数据库中，选择存取路径主要是指确定如何建立索引。例如，应把哪些域作为次码建立次索引，建立单码索引还是组合索引，建立多少个为合适，是否建立聚集索引等。

（3）确定数据的存放位置

为了提高系统性能，数据应该根据应用情况将易变部分与稳定部分、经常存取部分和存取频率较低部分分开存放。

（4）确定系统配置

DBMS 产品一般都提供了一些存储分配参数，供设计人员和数据库管理人员 DBA（DataBase Administrator）对数据库进行物理优化。初始情况下，系统都为这些变量赋予了合理的默认值。但是这些值不一定适合每一种应用环境，在进行物理设计时，需要重新对这些变量赋值以改善系统的性能。

2. 评价物理结构

数据库物理设计过程中需要对时间效率、空间效率、维护代价和各种用户要求进行权衡，其结果可以产生多种方案，数据库设计人员必须对这些方案进行细致的评价，从中选择一个较优的方案作为数据库的物理结构。

评价物理数据库的方法完全依赖于所选用的 DBMS，主要是从定量估算各种方案的存储空间、存取时间和维护代价入手，对估算结果进行权衡、比较，选择出一个较优的合理的物理结构。如果该结构不符合用户需求，则需要修改设计。

1.4.3.5 数据库实施

完成数据库的物理设计之后，设计人员就要用 DBMS 提供的数据定义语言和其他程序将数据库逻辑设计和物理设计结果严格描述出来，成为 DBMS 可以接受的源代码，再经过调试产生目标模式，然后就可以组织数据入库了，这就是数据库实施阶段。

数据库实施主要包括以下工作：

1. 定义数据库结构

确定了数据库的逻辑结构与物理结构后，就可以用所选用的 DBMS 提供的数据定义语言（DDL）来严格描述数据库结构。

2. 数据装载

数据库结构建立好后，就可以向数据库中装载数据了。组织数据入库是数据库实施阶段最主要的工作。对于数据量不是很大的小型系统，可以用人工方式完成数据的入库，其步骤为：

STEP 01 筛选数据

需要装入数据库中的数据通常都分散在各个部门的数据文件或原始凭证中，所以首先必须把需要入库的数据筛选出来。

STEP 02 转换数据格式

筛选出来的需要入库的数据，其格式往往不符合数据库要求，还需要进行转换。这种转换有时可能很复杂。

STEP 03 输入数据

将转换好的数据输入计算机中。

STEP 04 校验数据

检查输入的数据是否有误。对于中大型系统，由于数据量极大，用人工方式组织数据

入库将会耗费大量人力物力，而且很难保证数据的正确性。因此应该设计一个数据输入子系统，由计算机辅助数据的入库工作。

3. 编制与调试应用程序

数据库应用程序的设计应该与数据设计并行进行。在数据库实施阶段，当数据库结构建立好后，就可以开始编制与调试数据库的应用程序，也就是说，编制与调试应用程序是与组织数据入库同步进行的。调试应用程序时由于数据入库尚未完成，可先使用模拟数据。

4. 数据库试运行

应用程序调试完成，并且已有一小部分数据入库后，就可以开始数据库的试运行。数据库试运行也称为联合调试，其主要工作包括：

（1）功能测试，即实际运行应用程序，执行对数据库的各种操作，测试应用程序的各种功能。

（2）性能测试，即测量系统的性能指标，分析是否符合设计目标。

1.4.3.6　数据库运行和维护

数据库试运行结果符合设计目标后，数据库就可以真正投入运行了。数据库投入运行标志着开发任务的基本完成和维护工作的开始，但并不意味着设计过程的终结。由于应用环境在不断变化，数据库运行过程中物理存储也会不断变化，对数据库设计进行评价、调整、修改等维护工作是一个长期的任务，也是设计工作的继续和提高。

在数据库运行阶段，对数据库经常性的维护工作主要是由 DBA 完成的，它包括：

1. 数据库的转储和恢复

定期对数据库和日志文件进行备份，以保证一旦发生故障，能利用数据库备份及日志文件备份，尽快将数据库恢复到某种一致性状态，并尽可能减少对数据库的破坏。

2. 数据库的安全性、完整性控制

DBA 必须对数据库安全性和完整性控制负起责任。根据用户的实际需要授予不同的操作权限。另外，由于应用环境的变化，数据库的完整性约束条件也会变化，也需要 DBA 不断修正，以满足用户要求。

3. 数据库性能的监督、分析和改进

目前许多 DBMS 产品都提供了监测系统性能参数的工具，DBA 可以利用这些工具方便地得到系统运行过程中一系列性能参数的值。DBA 应该仔细分析这些数据，通过调整某些参数来进一步改进数据库性能。

4. 数据库的重组织和重构造

数据库运行一段时间后，由于记录的不断增、删、改，会使数据库的物理存储变坏，从而降低数据库存储空间的利用率和数据的存取效率，使数据库的性能下降。这时 DBA 就要对数据库进行重组织，或部分重组织（只对频繁增、删的表进行重组织）。



1.5 课后习题

1．试述文件系统与数据库系统的区别和联系。

2．试述层次、网状数据库的优缺点。

3．试述关系模型的完整性规则。

4．请简述 DML、DDL、DCL 的区别，并且分别列举出它们主要的 SQL STATEMENT。

5．将字段 COL1 赋值为 NULL 是否正确，为什么？

6．连接查询和子查询的区别是什么？如果两者在逻辑上等价，哪个效率更高？

7．静态 SQL 和动态 SQL 的区别是什么？

8．需求如下：

a．TBLB 定义为（COLA，DEC（10，0），COLB TIMESTAMP）。

b．对于 TBLA，在 TBLA 插入任何一条记录后，需要保证 TBLB 的 COLA 值加 1，同时在 COLB 中记录最后插入的时间戳。

请编写触发器实现以上需求。

9．需求分析阶段的设计目标是什么？调查的内容是什么？

10．什么是 E-R 图？构成 E-R 图的基本要素是什么？

第 2 章　主机DB2 基础

本章为主机 DB2 基础，主要介绍 DB2 产品的基本架构，包括 DB2 产品发展概述、DB2 数据基本结构、DB2 系统结构组成、DB2 系统运行环境、DB2 并发控制机制、应用程序的管理和开发、数据库安全控制等。通过对以上知识点的讲解，让读者能够对 DB2 系统的基础知识有一定的理解。

2.1　DB2 产品发展概述

DB2 是 IBM 公司数据管理产品线上最知名也是最成功的产品，在多家大型企业和行业得到非常广泛的应用，是主流数据库管理系统。DB2 产品自开始研制到目前已经过 31 年。

1968 年 IBM 公司在 IBM 360 计算机上研制成功了 IMS V1，这是业界第一个层次型数据库管理系统，也是层次型数据库中最著名、最典型的数据库管理系统。1970 年是数据库历史上划时代的一年，IBM 公司的研究员 E.F.Codd 发表了业界第一篇关于关系数据库理论的论文《A Relational Model of Data for Large Shared Data Banks》，首次提出了关系模型的概念。这篇论文是计算机科学史上最重要的论文之一，也奠定了他“关系数据库之父”的地位。1973 年，IBM 研究中心启动了 System R 项目，旨在探讨和研究多用户与大量数据下关系型数据库的实际可行性，这个项目在 DB2 发展史上有着重要的意义，它为 DB2 的

问世打下了良好的基础。

1974 年 IBM 的研究员 Don Chamberlin 和 Ray Boyce 通过 System R 项目的实践，发表了论文《SEQUEL：A Structured English Query Language》。论文中提出的 SEQUEL 语言是一套比关系微积分与关系代数更适合最终用户使用的非程序化查询语言，我们现在所熟知的 SQL 语言就是基于它发展起来的。

1982 年 IBM 发布了 SQL/DS for VSE and VM，这是业界第一个以 SQL 作为接口的商用数据库管理系统。1983 年千呼万唤始出来，IBM 在这一年发布了 DATABASE 2（DB2）for MVS（内部代号为“Eagle”）。1988 年 IDUG（国际 DB2 用户组织）组织成立，这标志着 DB2 的用户已经形成了相当的规模。

1993 年 IBM 发布了 DB2/2 和 DB2/6000，这是 DB2 第一次在 Intel 和 UNIX 平台上出现。

1994 年 IBM 发布了运行在 RS/6000 SP2 上的 DB2 并行版 V1，DB2 从此有了能够适应大型数据仓库和复杂查询任务的可扩展架构。

1995 年 IBM 发布了 DB2 Common Server V2，这是第一个能够在多个平台上运行的对象关系型数据库产品，并能够对 Web 提供充分支持。DataJoiner for AIX 也诞生在这一年，该产品赋予了 DB2 对异构数据库的支持能力。

1996 年 IBM 发布 DB2 V2.1.2，这是第一个真正支持 Java 和 JDBC 的数据库产品。

1997 年 IBM 推出用于 OS/390 V5 的适用于 Web 的 DB2，这是唯一一个支持多达 64 000 个并发用户和几百个千兆字节的数据库。

2000 年 IBM 推出 DB2 XML Extender，提供数据管理行业中第一个内置 XML 支持。

2008 年被称为 Viper 的 z/OS 操作系统下的新版 DB2 9 数据库软件，正式对公众发布上市。这款新版软件可以让主机更加容易地支持 DB2 数据库中的 XML 数据，同时也是第一个将常规关系数据和 XML 数据管理混合的数据管理系统。通过 DB2 9 能同时处理 XML 和 SQL，它支持一个数据库平台同时用于数据处理、文档处理和 SOA。对于经常接触 SQL 和表结构的人来说，XML 为结构化文档思想和新的查询技术打开了一扇门。现代的科技发展迅速，内容管理和面向 Web 的应用已经成为新的发展趋势，DB2 9 对 XML 良好的支持和管理，为新的分布式计算模型提供了支持，包括 Web 服务、网格服务和面向服务架构（SOA），表明 DB2 这一古老的产品时刻走在时代的前沿，并且不断拓展服务的领域和空间。



2.2　DB2 数据库对象

在 DB2 数据库中，表、视图、存储过程、触发器等具体存储数据或对数据进行操作的实体都称为数据库对象。数据库对象用于管理系统和用户的数据，可以让用户直接访问。

下面将对 DB2 的数据库对象进行逐一阐述。

2.2.1　DB2 数据库对象概貌

DB2 的数据库对象主要分为“databases”，“Storage Groups”，“Tablespaces”，“Tables”，“Indexes”，“Views”，图 2.1 全局概要描述了各个对象在 DB2 中的关系。

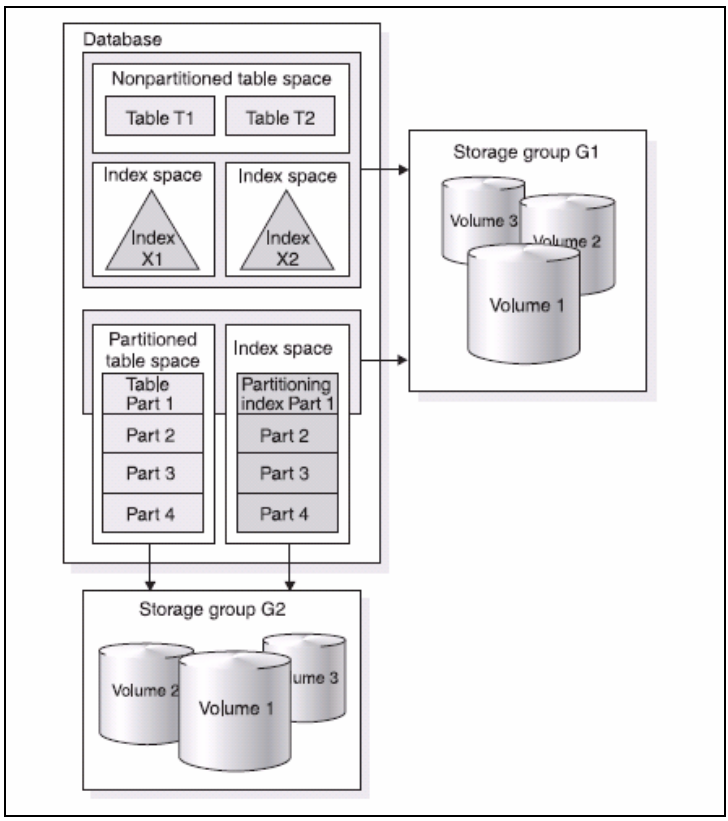


图 2.1　DB2 数据库对象概览

从图 2.1 中可以看出 DB2 的对象中有相互包含的关系，但从某种意义上讲包含也是一种层次性的结构。下面的章节将逐一对每一个对象做详细介绍。

DB2 的对象主要有以下几类。

- Database：一系列 DB2 数据库的数据结构和组织的集合，包括 Table 和相关的 Index，以及存放 Table 和 Index 的对应的 Tablespace、Indexspace。
- Storage Group：一组磁盘卷的集合，用于存放 Table 和 Index 对应的物理文件，在定义 Tablespace 和 Index space 时指定。
- Tablespace：一组定义在磁盘卷上的文件，用于存储 Table。
- Index space：一组定义在磁盘卷上的文件，用于存储 Index。
- Table：所有的数据在 DB2 database 中表示为 Table，Table 是包含相同列的记录的集合。存放固定的用户数据的 Table 称为基础表，存放临时数据的 Table 称为临时表。
- Index：在 DB2 内存放指向 Table 中的数据的一系列指针称为 Index，Index 与 Table 分开定义、分开存放。
- View：是存在于一个或多个 Table 内的数据说明的另一种形式；View 可以包含一张或多张基础表的全部或部分列数据。View 类似 Table 但本身并不存放数据。

2.2.2 Database

一个 Database 在设计时可以包含一个应用或一组相关联的应用对应的表。将关联应用的数据表定义在同一个 Database 中，可以使用户对同一类表执行操作时看作是一个集合，对集合进行操作和管理，这样操作和管理上都要简便和快捷得多。

在创建一个 Tablespace 或 Table 时，需要用户指定 Tablespace 或 Table 所在的 Database 名称，如果用户没有指定，DB2 会将 Tablespace 或 Table 建立在系统默认的 Database（DSNDB04）中。DSNDB04 是在系统安装 DB2 子系统时创建的。默认情况下所有的用户都有权限在 DSNDB04 中创建 Tablespace 或 Table，但系统管理员可以收回这一权限并且针对需求仅对特定用户授予必要的权限。

2.2.3 Storage Group

Storage Group，也称为存储组，是 DB2 的一个对象，是一组磁盘卷的集合，主要的用途是 DB2 为定义对象 Tablespace、Indexspace 分配和管理对应的物理空间。

磁盘卷通过 ICF CATALOG 或 VSAM CATALOG 联系起来。在 ICF CATALOG 中记录了所有建立在同一个 Storage Group 下对象的物理数据集。

以下是创建的例子：

```
CREATE STOGROUP SG1
  VOLUMNS（VOL1，VOL2）
  VCAT ALIASICF;
```

所有在 Storage Group 下管理的卷要写在 VOLUMNS 参数后面，在使用时，DB2 会顺序使用列在 VOLUMNS 参数下面的磁盘卷。

定义给一个 Storage Group 的盘卷并不仅供此 Storage Group 独享，一些非 DB2 的文件可以分配在此卷上，并且它也可以再定义给其他 Storage Group。一个 Storage Group 最多可以定义的磁盘卷是 133 个，且定义在同一个 Storage Group 下的磁盘卷必须是相同型号的硬件设备。

用户可以创建多个 Storage Group，如图 2-1 中所示，同一个 Database 中的对象可以存放在不同的 Storage Group。默认的 Storage Group 名称为 SYSDEFLT，在安装 DB2 子系统时创建，在创建 Tablespace、Index space 对象时没有显性地指定 Storage Group 时，DB2 系统会将它们创建在 SYSDEFLT 上。

由于 Storage Group 在管理物理磁盘方面的限制，在管理大量数据文件时，在海量数据库系统中使用 Storage Group 给系统管理员带来很大的工作量。但是在主机操作系统 V1.5 版本后，引入了一个 Data Facility Storage Management Subsystem（DFSMS）组件，该组件可以自动管理 DB2 使用和需要的所有文件。使用 DFSMS 管理数据库物理文件，将节省大量 DB2 系统管理员的工作负荷，也减少了由于存储空间的问题导致 DB2 不可用问题发生的可能。

在 Storage Group 定义中，如果在 VOLUMNS 参数中指（‘*’）而不是具体的卷标，就意味着存储管理交给 DFSMS 组件进行管理，Storage Group 不再管理物理空间。

2.2.4　Tablespace

Tablespace 由一系列的 VSAM 文件组成。文件格式是 VSAM 的线性类型（Linear Data Set，LDS）。Tablespace 在存储中以相同大小单元形式存储，单元称为页面（Page），Page 是 DB2 从磁盘做 IO 读写操作的最小单元。在定义 Tablespace 时用户可以指定 Page 的大小。

默认 Page 的大小是 4KB，也可以定义为 8KB、16KB、32KB 的 Page。当用户创建一个 Tablespace 时，需要指定使用的 Database 和 Store group，如果没有指定，DB2 会将 Tablespace 建立在系统默认的 Database 和 Store group 中。

DB2 定义 Tablespace 的类型有如下四类：

- 简单表空间（Simple Tablespace）
- 分段表空间（Segment Tablespace）
- 分区表空间（Partition Tablespace）
- LOB 表空间（LOB Tablespace）

2.2.4.1 简单表空间

简单表空间，也称为 Simple Tablespace，一个简单表空间可以装有一个或多个表，个数没有限制。定义在同一个简单表空间的多个表的记录行混存在同一个页面中。在一个简单表空间中放置多个表将影响数据的并行访问，影响数据的可用性，影响空间管理和装载数据 LOAD 工具的处理。当使用 REPLACE 参数的 LOAD 工具装载数据时，将所有的数据装入表空间，而不是只装入一个表的数据，因此可能将表空间中的其他表的记录数据覆盖。

在定义时使用简单表类型的表空间，并且在表空间中定义多个表，除非应用以预先定义好的顺序插入和读取数据，并且多个表的数据是以簇集的方式存放在同一页上，才能保证访问性能不下降。将多个表的数据行混合放置在同一个表空间的页面上，将对不以这种方式访问数据的所有查询、工具和应用产生不利的影响。在定义简单表空间这种类型之前，必须确认对数据的主要访问类型是这样预定义好的顺序，并且在数据插入时保证不改变。这对于用户来说有非常大的难度，因此目前在实际应用中基本上简单表空间的使用非常少，并且尽量保证同一个表空间中只定义一个表。

2.2.4.2 分段表空间

分段的表空间，也称为 Segment Tablespace，这类表空间也可以定义一个或多个表，表空间通过定义段（Segments）来划分成 4～64 个页面的独立空间，每一个段仅供分段表空间中定义的一个表使用，一个表可以占用一个或多个段。因此相同表空间中的多个表的记录行不会混存在相同的页面中。

对于大多数的 DB2 应用，建议使用分段的表空间，分段的表空间除可以提供简单表空

间所有的优势外，还可以提供以下优势：

（1）多个表可以定义在同一个分段的表空间里，不会出现使用简单表空间碰到的问题。表被存储在不同的段（Segment）中。由于不同表的数据行不再存储在同一个页面上，所以对同一分段的表空间中的表的并行访问不再是个问题。

（2）分段的表空间管理自由空间更为有效，对插入和可变长度行的更新将产生更少的开销。

（3）大批量删除操作更为有效，因为只是存储空间映射而不是数据本身被更新。在简单表空间中对表的大量行的删除，需要物理地读入和删除每一行。下面是一个大批量删除的例子：

```
DELETE FROM DSN8610.DEPT;
```

如果 DSN8610.DEPT 是在一个简单表空间中定义的，则所有的行要被读入和删除。如果它是在一个分段的表空间中定义，那么只有空间映射被更新，以指明所有这些行已被删除。

（4）被删除的表的对应空间可以立刻被回收，这就减少了对数据库进行重组维护的需要。

建议对所有的有多个表的表空间（不需要在一个页面上混合多个表的数据）都定义成分段的表空间。即使对每个表空间只定义一个表，分段的表空间特有的高效的空间使用所带来的高性能优势也是选择分段的表空间的因素。

定义分段的表空间的一个关键的参数是 SEGSIZE（段的大小），SEGSIZE 参数是一个表示分配给每一个段的页面数据的整数值，段的大小可以是任意从 4～64 之间的 4 的倍数。

当在分段的表空间中定义多个表时，需要在同一个表空间中存放大小尺寸类似的表。不要将大型的表和小型的表放在同一个分段的表空间中。在一个拥有巨大的段数的表空间中，定义小型的表将导致对 DASD 空间的浪费。

2.2.4.3　分区表空间

分区表空间，也称为 Partition Tablespace，这类表空间只能定义一个表，表空间被通过定义分区（Partition）来划分成多个称为分区的若干部分，每一个分区都对应一个单独的物理 VSAM 数据集。分区表空间的设计通过将大型的表中的数据分布在不同的物理磁盘设备上，来增强数据的可用性。

有两类的分区表空间：大型（LARGE）和非大型（non-LARGE）。定义为 LARGE 的

表空间允许每个 PARTITION 大小可以达到 4GB，进而允许一个表空间可以容纳超过 64GB 大小的数据。

分区表空间的分区个数，通过参数 NUMPARTS 指定，来表示表空间内包含多少个分区。目前在 DB2 V8 版本，可以支持最多定义 4096 个分区。

每个分区的最大容量，通过参数 DSSIZE 来指定，有效的 DSSIZE 值分为：1GB（1 gigabyte）、2GB（2 gigabyte）、4GB（4 gigabyte）、8GB（8 gigabyte）、16GB（16 gigabyte）、32GB（32 gigabyte）、64GB（64 gigabyte）。

当用户定义的 DSSIZE 大于 4GB 时，需要运行在具有 DFSMS 1.5 版本基础上的系统环境中，表空间的数据集必须和定义为扩展格式和扩展寻址能力的 DFSMS 数据集相关联。创建大于 4GB 的数据集需要使用 DFSMS 扩展寻址功能。IBM 用扩展寻址启动（EA-enabled）这个术语来定义能够支持扩展寻址能力的数据库。

分区表空间定义的 PARTITION 的个数依赖于页面的大小及 DSSIZE 参数的设置。整个分区表空间的大小依赖于定义的 PARTITION 个数总和及 DSSIZE 参数的设置。页面大小影响表空间的大小是因为它决定了一个分区表空间允许定义的 PARTITION 个数。

表 2.1 中整理了页面大小、DSSIZE、分区个数及表空间的关系。从表格中可以看出，对于非扩展寻址的数据库系统，最大允许的表空间为 16TB，对于支持扩展寻址的数据库系统，最大允许的表空间为 128TB。

表 2.1　分区表空间

类　型	页面大小	DSSIZE	Partition 个数	Tablespace 大小
Non-EA	4KB	4GB	4096	16TB
EA	4KB	1GB	4096	4TB
EA	4KB	2GB	4096	8TB
EA	4KB	4GB	4096	16TB
EA	4KB	8GB	2048	16TB
EA	4KB	16GB	1024	16TB
EA	4KB	32GB	512	16TB
EA	4KB	64GB	256	16TB
EA	8KB	1GB	4096	4TB
EA	8KB	2GB	4096	8TB
EA	8KB	4GB	4096	16TB
EA	8KB	8GB	4096	32TB

续表

类　型	页面大小	DSSIZE	Partition 个数	Tablespace 大小
EA	8KB	16GB	2048	32TB
EA	8KB	32GB	1024	32TB
EA	8KB	64GB	512	32TB
EA	16KB	1GB	4096	4TB
EA	16KB	2GB	4096	8TB
EA	16KB	4GB	4096	16TB
EA	16KB	8GB	4096	32TB
EA	16KB	16GB	4096	64TB
EA	16KB	32GB	2048	64TB
EA	16KB	64GB	1024	64TB
EA	32KB	1GB	4096	4TB
EA	32KB	2GB	4096	8TB
EA	32KB	4GB	4096	16TB
EA	32KB	8GB	4096	32TB
EA	32KB	16GB	4096	64TB
EA	32KB	32GB	4096	128TB
EA	32KB	64GB	2048	128TB

使用分区表空间的优点在于：

（1）每个分区可以指定不同的 Storage Group，以区分不同物理存储介质。

（2）每个分区被放在不同的 DASD 卷上，这将提高访问效率。

（3）分区表空间是唯一可以容纳大于 64GB 数据的表空间。64GB 是简单表和分段表空间最大容量。一个分区表空间可以容纳到 16TB 的数据（4096 个分区，每个分区容纳 4GB 的数据，4096×4=16TB）。一个支持扩展寻址的分区表空间可以容纳大到 128TB 的数据（2048 个分区，每个分区容纳 64GB 的数据，2048×64=128TB）。

（4）可以在分区一级上使用启动和停止命令，通过停止一个指定的分区，其他分区仍然可以使用，这样就提高了可用性。

（5）每一个分区都相对独立，对外的访问和操作都可以视为单独的对象，查询的 I/O、CPU 使多个引擎可以同时并行访问不同的分区，这通常会减少运行的时间，分区可以通过消除磁盘竞争使并行得到最优效率。

（6）对分区表空间的扫描可以跳过查询谓词中排除的分区，在表空间扫描时跳过分区

可以提高整个查询的性能。

（7）可以通过建立聚簇索引（Cluster Index）减少对数据的竞争。例如，如果表空间按部门分区，则每个部门的数据被放置在一个单独的分区文件上。每个部门在一个离散的物理数据集中，因而由于多个部门共存在一个页面上引起的部门间的访问竞争就减少了。注意，如果定义了非分区索引，非分区索引的数据仍然有竞争存在。因为非分区索引是有顺序的，如果用户在分区表上定义非分区索引，则分区级上的独立性所带来的数据库工具并行操作的好处将丧失。

（8）DB2 会为分区表空间的每一个分区创建一个单独的压缩字典。多个字典将会有更高压缩率。

（9）可以在分区表空间的分区一级上执行 REORG、COPY 和 RECOVER 等工具。这样工具执行的效率和并行度会大大提高，且在运行这些工具时由于分区的独立性和资源串行从而进一步增加了分区的可用性。这样对于系统维护和故障处理来说，在分区级别而不是整个表空间级别上处理，从效率和影响范围来说都要小得多。

（10）在 DB2 V8 版本后，对于表空间级别的锁也下放到分区一级，增大了数据更新的并发性。

使用分区表空间的缺点在于：

（1）每一个分区表空间只能定义一张表，这也不能说是缺点，仅是一个限制。

（2）每一个分区对应的物理数据集大小在定义时就已经确定，如果定义分区内存储的数据超过分区数据集定义的最大值，该物理数据集不会自动扩展，DB2 会报一个资源不可用的信息给用户，该分区标志为不可用。

（3）在创建分区索引前，作为插入表中数据依据的关键字的值的范围应当是一致的、稳定的。为了定义一个分区，值的范围必须在分区索引中通过编码来定义。这个范围应当能够根据应用使用数据时的访问需求将数据分布在各个分区上。

2.2.4.4 LOB表空间

大对象（LOB Large Object）是 DB2 用来管理非结构化数据的一种数据类型，存放的是如图形、图像、声频、视频或非常大的字符串等对象。一个包含了 LOB 字段的表需要建立在 LOB 表空间上。

在实际的存储和管理中，DB2 对 LOB 对象并不是用和其他 DB2 表中的数据一样的结构来存储的。包含 LOB 字段的表的定义在 LOB 表空间，称为基础表空间（Base Tablespace），

LOB 字段本身独立存储在单独表空间的一个附件表中（Auxiliary Table）。

2.2.5　Table

表是关系型数据库系统中最核心的概念，在关系型数据库中使用表来表示关系，并存储具体的数据。当用户在 DB2 中定义了一个表，就定义了一个有序的列的集合。

Table 的物理数据页面结构如图 2.2 所示，不同的 Tablespace 中的表的数据页面结构是相同的。

每一个表的数据页面都以一个 20 字节的头开始，页头记录了关于该页面的剩余部分的控制信息。在页的末尾有一个 1 字节的页尾，用于和页头中的某一字节进行一致检查，保证页面正确性。在每一页中紧接着页尾的位置是一个包含执行下一个可用的 ID 映射条目的指针序列。ID 映射是一系列连续的 2 字节行指针。对于表中的每一个数据行，都有一个行指针存在。每个行指针识别在数据页中的一个数据行的位置。每个数据页最大可以定义 255 个这样的指针。每个页中的最大行数通过 MAXROWS 参数定义，MAXROWS 参数表明了一个表空间页面能够存储的行数最大值，其默认值为 255。

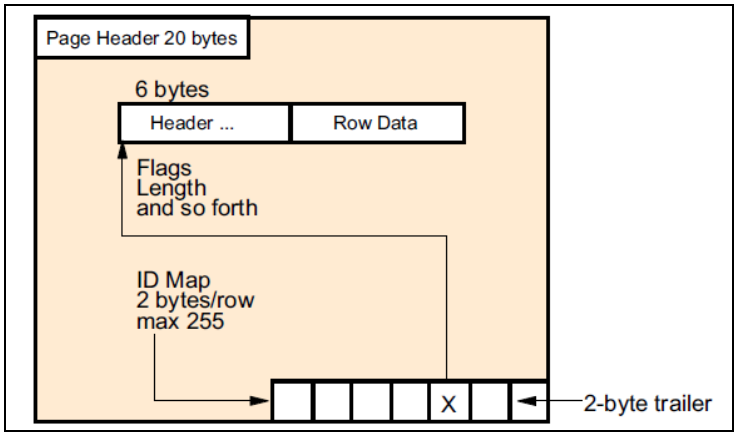


图 2.2　Table 的物理数据页面结构

每个数据页可以包含一个或多个数据行。每个行指针指向一个数据行，每个数据行包含了一个 6 字节的行头，用于管理数据行的状态。

每个表的数据存储在数据页的集合中，表中的行在数据页中不按特定的顺序存放，而且数据页也没有特定的组织顺序。

2.2.6 Index

在 DB2 内存放指向 Table 中的数据的一系列指针称为 Index，Index 与 Table 分开定义、分开存放。

索引的建立是加快查询速度的有效手段。用户可以根据应用环境的需要，在基本表上建立一个或多个索引，以提供多种存取路径加快查找速度。

一般说来，建立与删除索引由系统管理员 DBA 或表的属主（OWNER），即建立表的人负责完成。系统在存取数据时会自动选择合适的索引作为存取路径，用户不必也不能显式地选择索引。

索引一经建立，就由 DB2 系统使用和维护它，不需要用户干预。建立索引是为了减少查询操作的时间，但如果数据增删改频繁，系统会花费许多时间来维护索引，从而降低了查询效率。

每一个索引基于一张表的一个或多个列的值。用户定义了一个索引后 DB2 负责维护 INDEX 的结构，用户在必要的时候可以对索引进行必要的维护如重组、重建等操作，以保证索引的良好性能。索引对应的物理结构是 INDEX SPACE。每一个索引分别对应自己的 INDEX SPACE。

建立索引的主要目的是：

（1）提高性能，访问数据时使用索引通常比不使用索引要快。

（2）确保记录是唯一值。例如， 建立在员工登记表上的员工号列建立一个唯一索引，就确保了没有两个员工共享同一个员工号。通过建立唯一索引，可以确保索引列中不包含重复的值。只有当唯一性是数据本身的特征时，指定唯一索引才有意义。

除了性能表现上的差异，用户在访问表的数据时通常并不会意识到索引的存在。DB2 决定在访问表时是否使用索引。

2.2.6.1 索引的分类

索引类型分为聚集索引和非聚集索引。了解这两种类型索引的结构有助于创建合理的索引，提高查询速度。

（1）聚集索引

聚集索引指示表中数据行按索引键的排序次序存储，聚集索引对查找行很有效。只有当表包含聚集索引时，表内的数据行才按索引列的值在磁盘上进行物理上排序和存储。每

一个表只能有一个聚集索引，因为数据行本身只能按一个顺序存储。当查询指定了关键值的范围或者按照关键值的顺序访问数据行时，应考虑在对应的列上创建聚集索引。

聚集索引对于那些经常要搜索范围值的列特别有效。使用聚集索引找到包含第一个值的行后，便可以确保包含后续索引值的行物理相邻。例如，应用程序查询某一日期范围内的记录，如果在日期列上创建了聚集索引，则能够快速找到包含开始日期的行，然后检索表中所有相邻的行，直到到达结束日期。同样，如果对从表中检索的数据进行排序时经常要用到某一列，则创建聚集索引指示表中的行按照该行的键值顺序进行排序，避免每次查询该列时都执行排序，从而节省成本。例如，对于 ORDER BY 或 GROUP BY 子句中指定的列进行索引，可以使 DB2 不必对数据进行排序，因为这些行已经排序。这样可以提高查询性能。

当索引值唯一时，使用聚集索引查找特定的行也很有效率。

注意：聚集索引不适用于频繁更改的行，因为 DB2 必须按聚集索引的键值改变行的物理顺序存储，如果聚集索引列中的数据频繁修改，将导致聚集索引频繁修改，数据整行移动，消耗大量的系统资源。

（2）非聚集索引

非聚集索引具有完全独立于数据行的结构，数据行不按索引值的次序存储。在非聚集索引中，每个索引键都有指针指向包含该键值的数据行。

当用户需要使用多种方法查询数据时，非聚集索引非常有用。每张表只能有一个聚集索引，但是用户的查询条件是多样的。例如，员工表的 EmployeeID 列上已经创建了聚集索引，但是用户经常会使用员工的姓名、所在部门、出生日期等作为查询条件，所以应当考虑在这些列上创建非聚集索引，以提高查询效率。

如果创建索引时没有指定索引类型，默认情况下为非聚集索引。每个表最多可以创建 249 个非聚集索引。最好在唯一值较多的列上创建非聚集索引。

2.2.6.2　索引的物理结构

DB2 索引的物理结构是 B+ 树（平衡树）结构，B+ 树索引具有动态平衡的优点。为了快速查询，它将数据值进行了排序，被索引的值被存在一个颠倒的树结构中，如图 2.3 所示。

当值被插入索引或从索引中删除值时，树结构自动平衡，重新排列这个层次结构，使得从上到下的路径是一致的。通过使查询路径尽量短，这个重新排列操作减少了访问给定

值所需的时间。

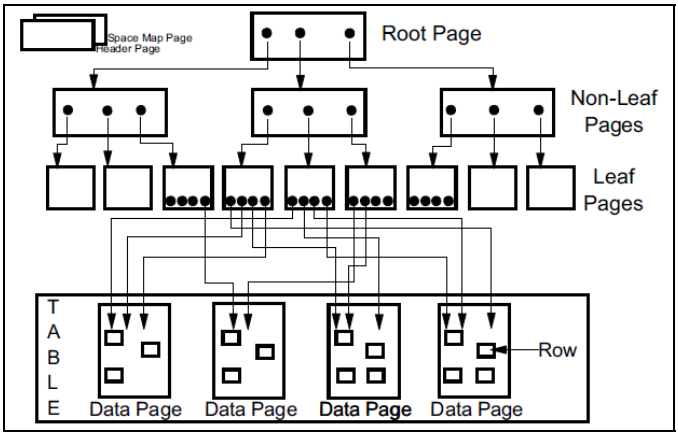


图 2.3　索引的物理结构

2.2.6.3　索引物理结构分类

索引的物理结构主要分三类：

- 根节点，每个索引只有一个根节点，存在于每个索引结构层次的最高级。
- 非叶子节点，是 B+树层次结构中的中间级别的节点，它可以不存在，如果存在，它们就包含了指向其他非叶子节点或叶子节点的指针，并不指向数据行。
- 叶子节点，是 B+树层次结构中最底层的节点，包含索引最重要信息，即指向表中数据行的指针。

叶子节点中的指针被称为“记录 ID”或 RID。每个 RID 是表空间页号和数据值行指针的组合，它们一起描述了数据值的位置。图 2.4 演示了索引叶子节点指向到表中数据行。

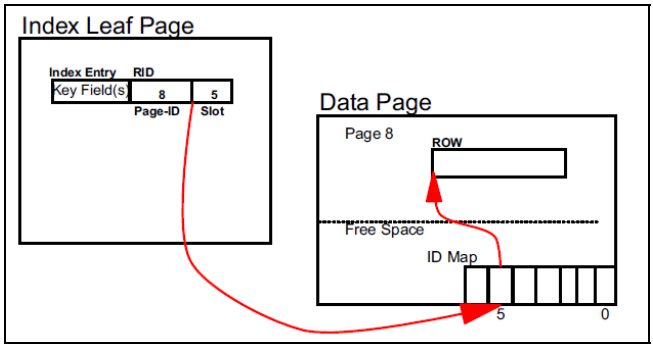


图 2.4　索引的叶子节点

2.2.6.4　使用索引的准则

业务规则、数据特征和数据的使用决定了创建索引的列。一般情况下，应当在经常被查询的列上创建索引。索引将占有磁盘空间，并且降低添加、删除和更新行的速度。但是索引对于数据检索速度的提高也比较明显，因此需要综合评估。

创建索引的基本准则是：

- 在搜索频繁的列上创建索引。
- 在包含唯一值较多的列上创建。

2.2.7　View

视图是从一个或几个基础表（或视图）导出的表。与基础表不同，它是一个虚表。建立视图，可以允许对终端用户屏蔽表的部分记录。视图可基于表的部分列或多个表的多个列组合定义。当用户创建一个视图时，DB2 会将视图的相关定义存储在 DB2 CATALOG 中，并不是存储视图对应的数据，这些数据仍存放在原来的基础表中。所以基础表中的数据发生变化，从视图中查询出的数据也就会随之改变。在访问视图时 DB2 根据 CATALOG 的信息，结合应用程序中 SELECT 语句到视图对应的物理表中产生具体的数据。从这个意义上讲，视图就像一个窗口，透过它可以看到数据库中自己感兴趣的数据及变化。

视图一经定义，就可以和基础表一样被查询、被删除。也可以在一个视图之上再定义新的视图，但对视图的更新（增、删、改）操作有一定的限制。

2.2.7.1　视图的操作

图 2.5 是建立视图的例子：

2.2.7.2　视图的作用

视图最终是定义在基本表之上的，对视图的一切操作最终也要转换为对基本表的操作。而对于非行列子集视图进行查询或更新时还有可能出现问题。既然如此，为什么还要定义视图呢？这是因为合理使用视图能够带来许多好处。

（1）视图能够简化用户操作

视图机制使用户可以将注意力集中在所关心的数据上。如果这些数据不是直接来自基本表，则可以通过定义视图，使数据库看起来结构简单、清晰，而且可以简化用户的数据查询操作。例如，那些定义了若干张表连接的视图，就将表与表之间的连接操作对用户隐

蔽起来了。换句话说，用户所做的就是对一个虚表的简单查询，而这个虚表是怎么得来的，用户无须了解。

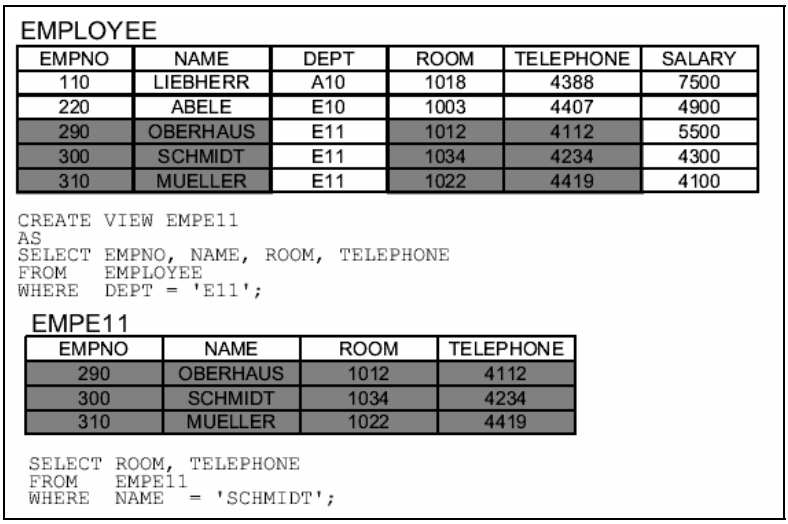


图 2.5　视图的建立

（2）视图使用户能以多种角度看待同一数据

视图机制能使不同用户以不同的方式看待同一数据。当许多不同种类的用户共享同一个数据库时，这种灵活性是非常重要的。

（3）视图对重构数据库提供了一定程度的逻辑独立性

数据的逻辑独立性是指当数据库重构造时，如增加新的关系或对原有的关系增加新的字段等，用户的应用程序不会受影响。在关系数据库中，数据库的重构往往是不可避免的，这时可以通过在新的关系上建立视图，将视图定义为用户原来的关系，保持用户的外模式不变，用户的应用程序通过视图仍然能够查找数据。

当然，视图只能在一定程度上提供数据的逻辑独立性，比如由于对视图的更新是有条件的，因此应用程序中修改数据的语句可能仍会因基本表结构的改变而改变。

（4）视图能够对机密数据提供安全保护

有了视图机制，就可以在设计数据库应用系统时，对不同的用户定义不同的视图，使机密数据不出现在不应看到数据的用户视图上。这样视图机制就自动提供了对机密数据的安全保护功能。例如 Student 表涉及全校 15 个院系的学生数据，可以在其上定义 15 个视图，每个视图只包含一个院系的学生数据，并只允许每一个院系的人只能查询和修改自己系的学生视图。

（5）适当地利用视图可以更清晰地表达查询

例如经常需要执行这样的查询“对每个学生找出他获得最高成绩的课程号”。可以先定义一个视图，求出每个学生获得的最高成绩：

```
CREATE VIEW VMGRAGE
AS
SELECT Sno, MAX（Grage）AS  Mgrade
FROM SC
GROUP BY Sno;
```

然后利用如下的查询语句完成查询：

```
SELECT SC . Sno , Cno
FROM SC,VMGRAGE
WHERE SC.Sno = VMGRAGE .Sno  AND  SC .Grade =  VMGRAGE . Mgrage ;
```

2.2.8　Synomas

如果你想要通过一个与表的本名不同的名字访问一张表，就需要创建一个 Synomas。Synomas 可以看作是给表或视图定义的别名。以下是一个实例。

真正的表名为 SYSIBM.SYSTABLESPACE，用户 STUDENT1 在访问 SYSIBM.SYSTABLESPACE 时必须写全表及它的 OWNER（SYSIBM），如下：

```
Select  *  from SYSIBM.SYSTABLESPACE where tbname='nthsjnl1';
```

如果 STDENT1 创建了一个 Synomas：

```
CREATE SYNONYM TS FOR SYSIBM.SYSTABLESPACE;
```

就可以在访问 SYSIBM.SYSTABLESPACE 时直接写：

```
Select  *  from TS  where tbname='nthsjnl1';
```

使用 Synomas，在避免全名引用时非常有效；

需要提示的是，Synomas 可以看成是一个私有的表的别名，只能被创建者使用。

2.2.9　Alias

Alias 同 Synomas 实现的功能基本相似，也是表或视图的别名，它可以看作是一个表的指针，通过 Alias 可以访问到另一张表的内容。指针本身也可以是一个虚表。与 Synomas

不同的是，Alias 可以看作是一个公有的别名，一旦定义，就可以被所有人访问，也可以引用到其他数据库系统。可以引用目前并不存在的对象；删除表或者修改表对 ALIAS 没有影响。以下是一个实例：

用户 BART 创建了一个表 EMP，用户 LISA 给 BART.EMP 创建了 ALIAS（AEMP），那么用户 LISA 和用户 BETH 都可以通过访问 AEMP 这个 Alias 访问到 BART 创建的表 EMP。

2.2.10 Trigger

触发器可以理解为一小段自动执行的代码，可以针对一些预先设定的事件实现自动的响应，事件主要是对 SQL 修改数据的语句（INSERT、DELETE、UPDATE）。触发器定义在指定的表上并且针对定义的表使用。触发器不能通过直接调用来执行，只能作为一个事件活动的结果被 DB2 触发执行，这个事件就其相关的表中的数据修改动作。当触发器创建后，只要它的触发事件发生（INSERT、DELETE、UPDATE），触发器就会执行，因此触发器是自动的、隐式执行的。使用触发器可以修改其他表，根据插入、删除、修改的值生成新的记录，可以执行 DB2 命令。

触发器的执行，可以通过在定义时指定 BEFORE 或 AFTER 来控制触发时刻，定义为 BEFORE，触发器会在触发事件发生之前执行，而定义为 AFTER，触发器会在触发事件发生之后执行。

2.2.11 Store Procedure

存储过程是一种特殊的数据对象，和触发器一样是过程化的、存储在数据库级的程序逻辑。在数据库系统中，存储过程是和其他对象之间并不存在“物理上”的联系，它能够访问或修改一个或多个表中的数据。在执行存储过程之前，必须直接显式地调用它，而不是像触发器一样由事件驱动。存储过程统一由 WLM 管理，运行在一个单独的地址空间 SSIDWLM 中，这样能够有效地隔离用户代码和 DB2 代码，防止存储过程引起整个 DB2 子系统崩溃。

DB2 的存储过程是很有用的应用程序，使用它的主要原因是可重用性、一致性、数据完整性、易维护性、高性能和安全。当多套环境都需要相同的程序代码时不用将所有的代码部署到多个环境，通过调用存储过程方式就可以轻松实现；对于相同的请求调用相同的存储过程，可以实现确保使用相同的逻辑功能；使用存储过程可以减少维护代码的工作量，

同时可以编写存储过程来支持数据库完整性约束。使用存储过程还有一个重要的原因是可以提高性能。在 B/S 环境中，通过调用存储过程而不是程序，在一次调用中就可以实现多个 SQL 的执行和输出结果，能够有效地降低网络通信量。另外通过设置存储过程，可以控制用户对不必要的对象进行访问。

2.2.12　创建数据库对象的方法

使用 SQL 中的 DDL 语句可以创建、修改、删除数据库对象，具体的语句语法可以参考具体的 SQL 语言语法书，本节不做赘述。

2.2.13　OWNER的概念

OWNER，也可翻译为属主。每一个 DB2 对象都有自己的属主，属主被记录在 DB2 的系统表中，当用户在创建对象时需要显性地指明对象的属主，如果没有指明，DB2 系统会默认将创建对象的用户当前 ID 作为该对象的属主。

在访问数据库对象时，如果用户没有指定对象的属主，DB2 系统会默认访问的对象属主是当前用户 ID。

属主用户对自己的对象有最高权限，可以使用、修改、删除自己拥有的对象，同时属主还可以决定谁可以访问自己拥有的对象，对其他人进行授权，如图 2.6 所示。

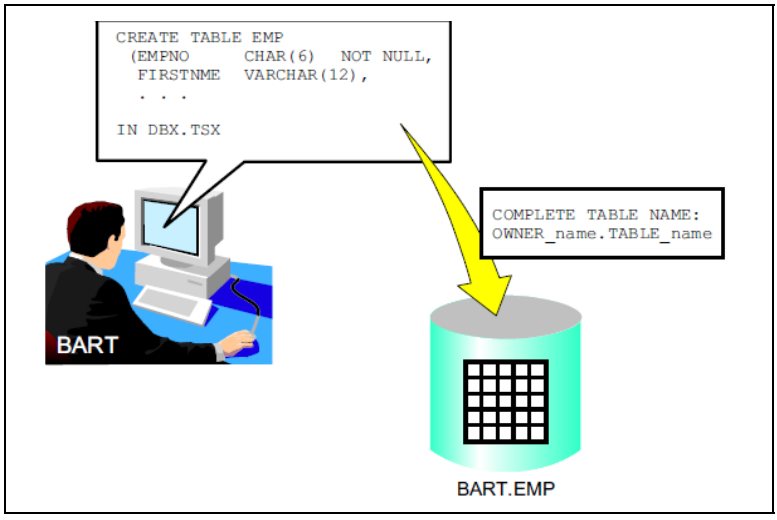


图 2.6　OWNER 的概念

2.2.14 数据库对象的命名规范

数据库对象的命名规范如图 2-7 所示。

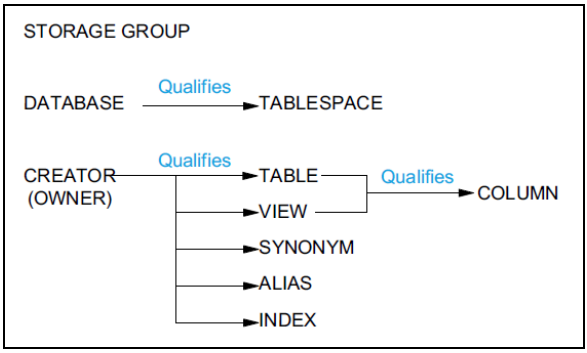


图 2.7 数据库对象的命名规范

Database 和 Storage Group 的名称在一个 DB2 系统中必须是唯一的。其他对象定义在同一个 Qualify 下的名称也必须是唯一的。

对象的命名必须由（A～Z，#，＄）字符开始，接下来以（A～Z，#，＄，0-9）等字符进行命名。

其中 Database、Tablespace、Storage Group、Creator 最多只能是 8 个字符，其他对象最多可允许 128 个字符。

一个完整对象的名称有三段 qualify，LOCATION.OWNER.Tablename，当用户在本地的 DB2 系统上访问表时无须指明 LOCATION。

2.2.15 数据库对象对应VSAM数据集的命名规范

DB2 创建的对象对应的物理数据集在系统中为线性 VSAM 文件，同一套数据库系统中建立的数据库对象对应的物理数据集都受操作系统 VSAM CATALOG 的统一管理。

对象的物理数据集有统一的命名规范：

```
Vcat.DSNDBx.Dbname.Tsname.m0001.Annn
```

- Vcat：表示最高层级的 Qualifier（来自 STOGROUP 定义中 VCAT 参数）。
- X：C 是对应的 Cluster，D 是对应的数据实体。

- Dbname：对应 Database 名称（最长 8 个字符）。
- Tsname：对应 Tablespace 的名称（最长 8 个字符）或者 Index space 名称的前 8 个字符。
- M：可以是 I 或 J。
- N：数据集或分区编号，从 001 开始。

其他部分都是 DB2 子系统保留关键字。

2.2.16　查询数据库对象的方法

DB2 自动将创建好的对象定义信息存放在 DB2 的系统表（编目表）中，当删除一个对象时，该对象在系统表中对应的信息也自动一并删除。查询数据库对象的信息，可以通过查询对应 DB2 系统表的对应表。

图 2.8 所示是对象定义和对应的系统表之间的关系。

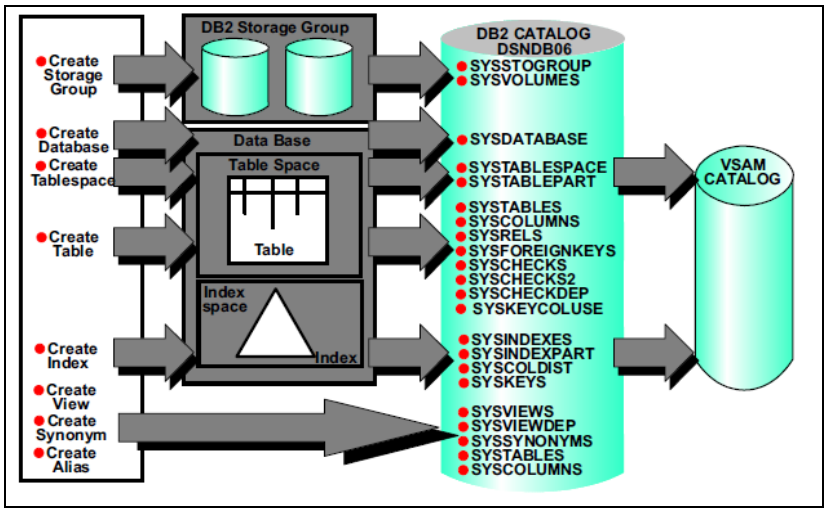


图 2.8　对象定义与系统表的关系

2.3　DB2 系统结构组成

DB2 管理的结构主要分为两大类：数据结构和系统结构，数据结构用于管理系统和用

户的数据，可以由用户访问，系统结构只能由 DB2 自己控制，不对用户开放。

下面将首先对 DB2 的系统结构进行阐述。

2.3.1 DB2 系统结构概貌

DB2 的系统结构主要分为如图 2.9 所示几个重要的部分。

每一个构件之间都有密切的关系，以下将分别对 DB2 的系统结构做逐一解释。

2.3.2 DB2 Catalog

DB2 Catalog 是所有 DB2 对象的知识库，保存了数据库对象的定义信息，可以称为 DB2 子系统的大脑。DB2 Catalog 由一系列 Tables 组成，存放在系统数据库 DSNDB06 内。它们记录了定义在 DB2 下的所有内容的相关数据信息，如：Table、View、Index 的创建、更改、删除等。这些 Tables 的命名都是以“SYSIBM.SYS”开头的，可以使用 SQL 语言对其访问。DB2 Catalog 中包括的 TABLESPACE 有：SYSCOPY,SYSPLAN,SYSPACKAGE,SYSDBASE 等。

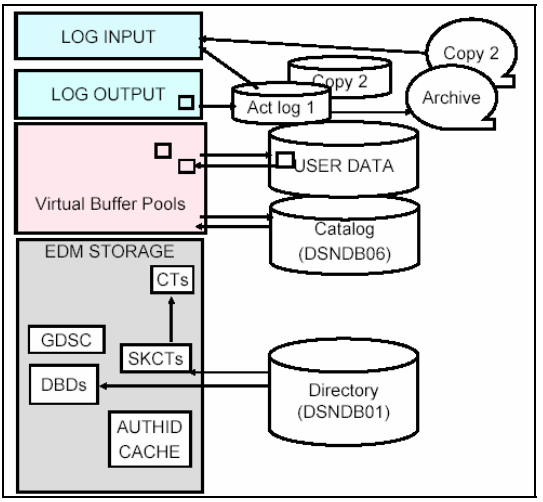


图 2.9 DB2 系统结构概貌

DB2 Catalog 中包含的信息主要有以下几类。

1．对象

STOGROUPS、Database、Tablespace、Partition、Table、Column、Index、View、Alias、

Key、Function、Trigger、Plan、Package 等定义信息及物理组织信息。

2．安全性

数据库各类权限的授予情况，如数据库安全性、计划和包的安全性、系统级别的权限、在表和视图上的安全性、资源特权等信息。

3．实用程序

各类 Utility 执行情况记录，如 IMGCOPY、LOAD、RUNSTATS 及对象组织有效性信息。

4．分布

为了数据分布和 DRDA 使用，DB2 子系统的连接方式。

DB2 Catalog 中数据信息的更新，主要有以下途径：

- 当使用 SQL 的 DDL 语句创建 DB2 对象（如 Database、Tablespace、Table）时，DB2 自动将对象相关的描述信息存放到 DB2 Catalog 的对应表中。
- 对于安全性 SQL 语句也是一样，当执行 GRANT 或 REVOKE 语句时引起的信息变化也会在语句执行完毕后被同步更新到 Catalog 的对应表中。
- 通过执行 RUNSTATS 等实用程序，对于 DB2 对象的物理组织信息也会自动收集到 DB2 Catalog 表中。

DB2 Catalog 表和 DB2 子系统是绑定在一起的，没有 DB2 Catalog 的 DB2 就不能定义任何对象。对于绝大多数的 DB2 Catalog 表而言，不能使用标准的 SQL 语句来修改它，一旦修改，有可能带来系统方面的不一致进而导致 DB2 子系统异常。

2.3.3　DB2 Directory

DB2 Directory 是 DB2 子系统的另一个非常重要的系统表，保存了 DB2 操作方面详细的技术信息。DB2 Directory 只能供 DB2 子系统内部使用，不能使用 SQL 访问。

DB2 Directory 由 5 张 Table 组成，存放在系统数据库 DSNDB01 内。但是这 5 张表不是真正的 DB2 表，它们被认为是一个结构，这些结构控制 DB2 任务，并包含 DB2 使用的复杂的控制结构。

DB2 Directory 包括的 Table 如下。

1．SCT02

SCT02 结构保存了 DB2 应用计划的框架游标表（SKCT）。这些 SKCT 中包含了实施 DB2 优化器确定的访问路径逻辑的指令。

在执行 BIND PLAN 命令时创建 SKCT，并存放在 SCT02 结构中。在执行 FREE PLAN 命令时使得对应的 SKCT 从 SCT02 结构中删除。当一个 DB2 程序被运行时，DB2 装载 SKCT 到一块内存区域（EDM POOL），以启动在应用程序中嵌入的 SQL 的执行。

2．SPT01

SPT01 结构保存了 DB2 应用计划的框架包表（SKPT）。这些 SKPT 中包含了实施 DB2 优化器确定的访问路径逻辑的指令。

在执行 BIND PACKAGE 命令时创建 SKPT，并存放在 SCP01 结构中。在执行 FREE PACKAGE 命令时使得对应的 SKPT 从 SPT01 结构中删除。对于一个基于一个计划的 DB2 程序（带有包列表）被运行时，DB2 装载 SKPT 和 SKCT 到一块内存区域（EDM POOL），以启动在应用程序中嵌入的 SQL 的执行。

3．DBD01

数据库描述符（DBD），被保存在 DBD01 的结构中。一个 DBD 是所有 DB2 对象（这些对象都属于同一个 Database 内）的内部描述。DB2 使用 DBD 作为这些对象在 DB2 Catalog 中保存的信息的一个有效表示。DB2 不通过访问 DB2 Catalog 来获得 DB2 对象信息，而是访问存储在 Directory 中的 DBD，这会提高整体效率。

4．SYSUTILX

DB2 子系统监视所有联机 DB2 Utility 的执行情况。所有启动的 DB2 Utility 的状态信息被维护在 SYSUTILX 结构中。当每个 Utility 执行时，执行的步骤和状态信息都被记录下来。当 Utility 正常结束或被终止时，就会自动从 SYSUTILX 中清除关于该 Utility 的所有信息。

5．SYSLGRNX

记录表空间的更新、DB2 日志的 RBA 范围信息，用于在 DB2 需要执行恢复时，可以有效定位需要的日志，并快速地识别需要恢复的日志的位置。

2.3.4　默认数据库

默认数据库 DSNDB04 在安装 DB2 子系统时定义，如果用户建 Table 或 Table Space 时没有指定 Database，系统将自动指定默认的 Database—DSNDB04，其默认的 Buffer pool 为 BP0，默认的 Storage Group 为 SYSDEFLT。

2.3.5　work file database

work file database 存放在系统数据库 DSNDB07 内，用于进行 SORT 的临时工作区。

2.3.6　Active and Archive log

记录 DB2 数据改变及重大事件发生的日志，用于故障恢复。DB2 将每个日志记录写到 DASD Dataset 内，叫 Active log。一般 Active log 有 4 个，且为双备份，当一个 Active log 写满后，DB2 将自动将 Active log 中的内容写到 DASD 或磁带的 Dataset 内，叫 Archive log。

一个程序更新 DB2 的数据时，当程序执行完 COMMIT 或 ROLLBACK 后，首先会将更新的数据存放在数据缓存中，而将更新后的数据写出到磁盘是以异步的方式，在 COMMIT 或 ROLLBACK 后的某一时间段进行。在 COMMIT 或 ROLLBACK 执行的同时，同步执行的操作只有向 DB2 系统日志数据集写入的操作。

系统日志数据集是记录了记录数在执行更新前后的数据值，以保证 DB2 系统可以在系统发生任何情况时都执行恢复。

如图 2.10 就是数据库系统日志工作示意图。

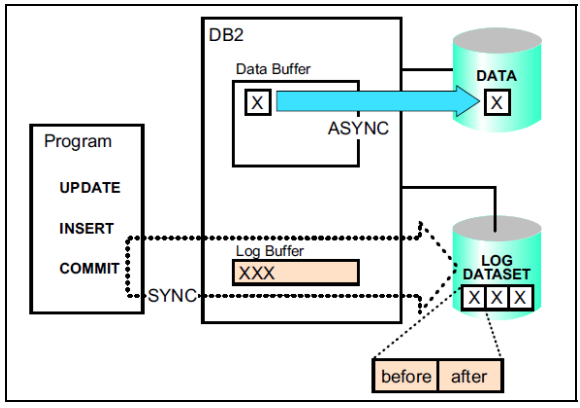


图 2.10　DB2 系统日志工作示意图

2.3.7 Bootstrap Data Set（BSDS）

The Bootstrap Data Set（BSDS）是一个 VSAM Key-Sequenced Data Set（KSDS）。它包含了 DB2 的关键信息；如：所有 Active 和 Archive log 数据集的详细目录（RBA 范围、Active log 的状态）、所有当前 DB2 的 checkpoint（用于重启）的状态、DDF 通信记录（包括 DB2 的域名、VTAM 的 LU 名）等。

BSDS 可以看作是记录 DB2 Active 和 Archive 日志的表，也可以描述为物理日志的结构和索引。它帮 DB2 决定哪一段日志文件可以用来恢复某一个特殊的日志点。日志文件包含非常简单的内容，它就是一串字节，每一个字节都有一个唯一的可以定位的地址，被称为 RBA（相对字节地址）。DB2 的日志通过将一串串字节的数据存在一个物理的文件集中来实现，而每一个物理文件集记录的内容记录在 BSDS 中。当某一个 Active 日志文件写满时，DB2 系统会触发 OFFLOAD 过程，OFFLOAD 过程自动创建一个新的 Archive 文件，将写满的那个 Active 日志文件的内容复制到新创建的 Archive 文件中，复制完成后，这一个 Active 日志文件又可以被新的日志记录覆盖。BSDS 和日志都是 DB2 系统非常重要的文件，通常建议对于 BSDS 和日志文件都采用双写机制，即每个 BSDS 和日志文件都是双份的，以防止其中一份文件的损坏导致整个 DB2 系统出现故障。

图 2.11 是 BSDS 和日志工作示意图。

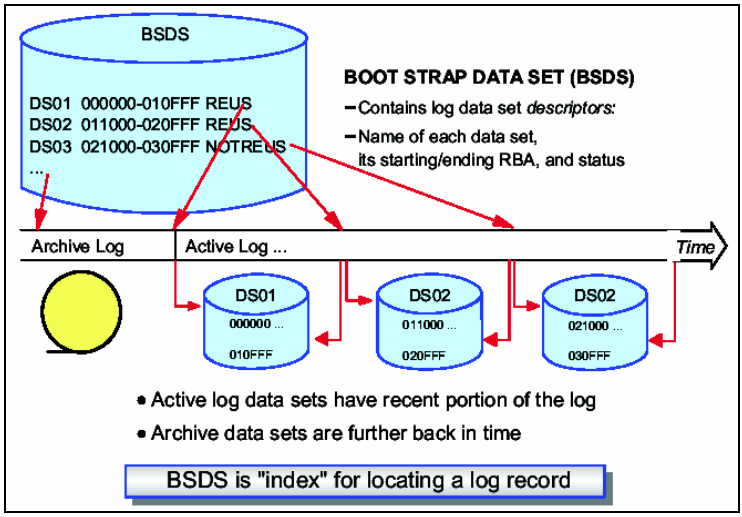


图 2.11 BSDS 和日志工作示意图

2.3.8　Buffer pool

Buffer pool 也称为虚拟 Buffer pools（缓冲池），是 Table Space 或 Index 的 pages 的临时虚拟存储区域，实际位于 DBM1 地址空间的 2G 线上区域。DB2 通过定义 Buffer pool 来降低 I/O 成本的。DB2 在 Buffer pool 中存放最近访问过的表和索引页面，当应用程序需要访问数据时，如果该数据行对应的页面存放在 Buffer pool 中就可以避免从磁盘上读取，减少了物理 I/O 的次数，能够显著提高性能。

DB2 提供下列缓冲池：

- 50 个 4KB 页面的缓冲池（命名从 BP0 到 BP49）。
- 10 个 8KB 页面的缓冲池（命名从 BP8K0 到 BP8K9）。
- 10 个 16KB 页面的缓冲池（命名从 BP16K0 到 BP16K9）。
- 10 个 32KB 页面的缓冲池（命名从 BP32K0 到 BP32K9）。

缓冲池的大小是按照页为单位指定的。

用户定义表空间时指定对应的 Buffer pool，在访问该表空间时 DB2 系统就会按照定义将表空间的记录装载到定义的 Buffer pool 中。

2.3.9　EDM pool

EDM pool 是一块 DB2 子系统特定的内存区，实际位于 DBM1 地址空间的 2G 线下区域。EDM pool 内存区由三部分组成，每一部分都是独立的内存区域。可以通过指定 DSNZPARM 参数来设定每一块内存区的大小。

EDM pool 的组成分为：

（1）存放应用程序相关的信息

- DB2 应用计划的框架游标表（SKCTs）。
- 活动的游标表（CTs），或 SKCTs 的拷贝。
- DB2 应用计划的框架包表（SKPTs）。
- 活动的包表（PTs），或 SKPTs 的拷贝。
- DB2 应用计划执行权限信息。

（2）EDM DBD CACHE

存放数据库对象定义的 DBD 信息。

（3）EDM Statement CACHE

包含动态 SQL 语句的框架信息。

在 DB2 V8 版本以前，以上三部分内容都存放在一个 EDM Pool 中，通过划分独立的空间分别存放。当 DB2 子系统升级到 V8 版本后，这三部分内容划分成三块独立的内存空间，分别为 EDM pool、EDM DBD CACHE、EDM Statement CACHE，并且 EDM DBD CACHE、EDM Statement CACHE 都由 2G 线下移到 2G 线上空间。

2.3.10 RID pool

RID pool 是进行 RID SORT 所需的存储空间。一般对于采用 LIST PREFETCH 的访问方式的应用程序将会导致需要进行 RID 的 SORT，如果 RID Pool 不够大，DB2 系统会自动由 LIST PREFETCH 变为 SEQUENTIAL PREFETCH 导致访问效率的恶化。当应用程序的 LIST PREFETCH 增加时，也要相应增加 RID pool 的大小。

2.3.11 DSNZPARM

DB2 子系统各项环境参数和系统配置，都是通过在 DB2 系统启动时指定的一系列系统参数设置的。这些系统参数通常被称为 DSNZPARM 或简称为 ZPARM。

DSNZPARM 中定义了许多 DB2 的关键参数，它是由一系列 DSN6XXX 的宏参数组成的，通过执行 DSNTIUZ 这个 CLIST，实现编译链接，生成一个模块，用于在 DB2 启动时装载。

在 DB2 6 版本以前，绝大部分系统参数的修改只能通过重新编译 DSNZPARM，并重新启动 DB2，让编译后的 ZPARM 重新 LOAD 到 DB2 系统内存中才能生效。这种做法影响了 DB2 系统的连续运行，降低了系统可用性。在 DB2 7 版本后，可以支持部分系统参数的联机生效，通过引入一个 SET SYSPARM 的命令，实现将新编译生成的 DSNZAPRM 模块联机装载到正在运行的 DB2 系统中，大大提高了 DB2 的可用性。

如图 2.12 所示是一段 DSNZARM 的内容。

```
DSN6LOGP   OUTBUFF=4000K,WRTHRSH=20
       TWOACTV=YES,TWOARCH=YES,
       MAXARCH=500
DSN6ARVP   ARCPFX1=DSNCAT.ARCHLOG1,
       ARCPFX2=DSNCAT.ARCHLOG2,
       TSTAMP=NO,BLKSIZE=28672,
       UNIT=TAPE,COMPACT=NO,CATALOG=NO,
       ARCWTOR=YES,ARCWRTC=(1,3,4)
DSN6SYSP   ROUTCODE=1,CHKFREQ=50000,RLF=NO,
       IDFORE=100,IDBACK=20,CTHREAD=70,
       AUDITST=NO,SMFACCT=1,SMFSTAT=YES,MON=NO
DSN6SPRM   RESTART,ALL,AUTH=YES,
       DSMAX=2000,CATALOG=dsncat,
       SITETYPE=LOCALSITE,EDMPOOL=1900,
       IRLMAUT=YES,IRLMPRC=IRLMPROC,
       IRLMSID=IRLM,IRLMRWT=60,
       SYSADM=SYSADM,SYSADM2=SYSADM,
       SYSOPR1=SYSOPR,SYSOPR2=SYSOPR,
       DEFLTID=IBMUSER,RELCURHL=YES
DSN6FAC    DDF=NO
```

图 2.12　DSNZARM 示例



2.4　DB2 系统运行环境

本节对组成 DB2 系统的不同组件和与之一起工作的 z/OS 系统环境进行简单的描述。DB2 像 z/OS 的一个子系统一样运行。DB2 Utilities 运行在 BATCH 环境下，访问 DB2 资源的应用可以运行在 BATCH、TSO、IMS、CICS 环境下。IBM 提供了 Attachment Facility，将 DB2 连接到这些环境。

2.4.1　DB2 系统的地址空间

为了能够正常运行，DB2 系统需要运行以下几个地址空间：ssnmMSTR、ssnmDBM1、ssnmIRLM、ssnmDIST。

2.4.1.1　ssnmMSTR

这个地址空间是必需的，它的主要功能是提供子系统服务。它主要控制如下的操作：

- 控制访问 DB2 的用户数量。
- 管理 DB2 监控和启停的组件和 DB2 命令处理。
- 记录数据改变前后的日志。
- 保护输入和输出日志缓存。

2.4.1.2 ssnmDBM1

这个地址空间是必需的，它的主要功能是提供数据库服务管理。这个地址空间的主要部件包括：

- Relational Data Services（RDS）Stage 2 SQL 的处理。
- Data Manager（DM）Stage 1 SQL 的处理。
- Buffer Pool Manager。
- Buffer Pool Areas。
- RID Pool Area。
- Sort Spaces。
- Work Space。
- EDM Pool（Environmental Descriptor Manager）。

2.4.1.3 ssnmIRLM

这个地址空间也是必需的，它的主要功能为锁管理：锁的建立和释放及死锁的检测。每个 DB2 系统都有一个自己的锁资源管理器，不能在多个 DB2 系统之间共享 IRLM。IRLM 和 DB2 一起对数据的访问进行串行化，当应用、命令、工具程序等都试图访问一份同样的数据时，DB2 向 IRLM 申请锁以确保数据的一致性。

2.4.1.4 ssnmDIST

这个地址空间不是必需的，它的功能是控制与 VTAM 之间的接口，管理分布式的数据访问。当有分布式数据访问需求时启动。

2.4.2 DB2 Attachment Facilities

本节将描述使用 z/OS 环境下的哪些 Attachment Facilities 可以启动一个 DB2 会话。你可以使用 ODBC、JDBC 和 SQLJ 等接口，从位于其他系统上（例如 Windows 或者 UNIX）的客户端来启动一个 DB2 会话。

一个 Attachment Facility 提供了 DB2 和其他环境的接口。图 2.13 展示了与 DB2 之间存在接口的 z/OS Attachment Facility。

z/OS 环境包括：

- WebSphere

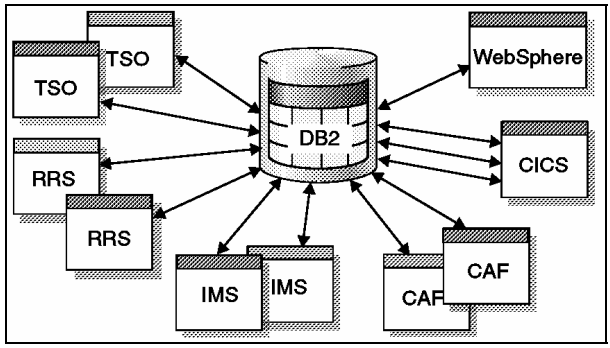


图 2.13　DB2 Attachment Facilities

- CICS（Customer Information Control System）
- IMS（Information Management System）
- TSO（Time Sharing Option）
- Batch
- z/OS Attachment Facilities 包括：
- CICS
- IMS
- TSO
- CAF（Call Attachment Facility）
- RRS（Resource Recovery Services）

在 TSO 和 Batch 环境下，可以使用 TSO、CAF、RRS Attachment Facilities 来访问 DB2。

2.4.2.1　WebSphere

与 DB2 集成在一起的 WebSphere 产品包括：WebSphere Application Server, WebSphere Studio, and Transaction Servers & Tools。在 WebSphere 环境下，可以使用 RRS Attachment Facility。

2.4.2.2　CICS

客户信息控制系统（CICS，Customer Information Control System）Attachment Facility 可以从 CICS 系统中访问 DB2。在 DB2 系统启动之后，可以从 CICS 的终端上对 DB2 进行操作。对 DB2 和 CICS 的启动和停止是互相独立的，可以在任何时间建立和终止它们之间

的连接。也可以选择让 CICS 自动连接到 DB2。

CICS Attachment Facility 提供了一些运行在 CICS 环境下的应用程序访问 DB2 数据。因此，CICS 应用既可以访问 DB2 数据，也可以访问 CICS 数据。在遇到系统故障的情况下，CICS 协调 DB2 和 CICS 数据的恢复。

编写 CICS 命令级程序的程序员可以使用同样的数据通讯编码技术，来编写访问 DB2 数据的应用程序的数据通信部分。编程的过程中，只是数据库部分需要改变。对于数据库部分，程序员使用 SQL 语句来获取或更改位于 DB2 表中的数据。

对于一个 CICS 终端用户而言，同时访问 CICS 和 DB2 数据的应用程序与只访问 CICS 数据的应用程序是相同的。

通过与 CICS 之间协调可恢复的资源，DB2 支持跨产品的编程。因此，CICS 应用既可以访问 CICS 管理和控制的资源，也可以访问 DB2 数据库。

在 CICS MRO（Multi-Region Operation）环境下，每个 CICS 地址空间都可以使用自己的 attachment 连接到 DB2 子系统。一个独立的 CICS 地址空间一次只能连接到一个 DB2 子系统。

一个授权的 CICS 终端操作员可以通过执行 DB2 命令来对 Attachment Facility 和 DB2 本身进行监控和管理。授权的终端操作员也可以启动和停止 DB2 数据库。

虽然通过 CICS 可以实施 DB2 的部分功能，TSO Attachment Facility 和 ISPF 还是需要的，以利用 DB2 提供的联机功能对系统进行安装和客户化。同时，还需要利用 TSO attachment 来绑定应用计划和应用包。

2.4.2.3 TSO

绑定应用计划与包和执行 DB2 提供的几个联机功能时，需要使用时间共享选项（Time Sharing Option—TSO）Attachment Facility。

使用 TSO Attachment Facility，使得用户既可以在前台，也可以在后台访问 DB2。使用 TSO 终端，可以在前台访问 DB2；通过从批量作业中调用 TSO 终端监控程序（Terminal Monitor Program，TMP），可以在后台访问 DB2。

下面的两个命令处理器是可以利用的。

（1）DSN 命令处理器：作为 TSO 的命令处理器运行，使用 TSO Attachment Facility。

（2）DB2 Interactive（DB2I）：由 ISPF 面板组成。ISPF 调用 DSN 命令解释器，可以建立与 DB2 之间的交互式连接。使用 DB2I 的面板可以以交互的方式执行大多数 DB2 的功能，

例如运行 SQL 语句、命令和 Utilities。

在 DSN 命令解释器下执行的 DB2 命令受到 TSO 定义的命令尺寸的限制。TSO 允许授权的 DB2 用户或者作业创建、修改、维护数据库和应用程序。通过在 TSO 终端发布命令，是在前台调用了 DSN 命令解释器。在后台，首先在批量作业中调用 TMP，然后将 SYSTSIN 中的命令传给 TMP。

在 DSN 命令解释器运行后，就可以执行 DB2 命令或者 DSN 子命令。但是在 DSN 命令解释器中，不能执行启动 DB2（-START DB2）的命令。如果 DB2 没有启动，DSN 命令解释器就不能建立与 DB2 之间的连接；为了能够将命令传给 DB2 进行处理，DSN 命令解释器需要建立和 DB2 之间的连接。

2.4.2.4　CAF

大多数 TSO 应用必须使用 TSO Attachment Facility，因为要通过它调用命令解释器。同时，DSN 命令解释器和 TSO 提供了下面的服务：自动连接到 DB2、attention key support、错误码向错误消息的转换等。然而，当使用 DSN 服务的时候，应用必须在 DSN 的控制下运行。

对于在会话过程中需要严格控制的 TSO 和批量应用，Call Attachment Facility（CAF）提供了另外一种连接方式。通过使用 CAF 提供的连接功能，使用 CAF 的应用可以显式地控制与 DB2 之间连接的状态。

2.4.2.5　RRS

z/OS 资源恢复服务（Resource Recovery Service，RRS）是 CAF 的一种较新的实现方法，拥有更多的功能。在 z/OS 系统中，RRS 有这样一种特性：协调可恢复资源的提交处理。对于使用了 DB2 提供的 RRS Attachment Facility 的应用来说，DB2 支持使用这些功能。使用 RRS attachment，在一个单一的交易中就可以访问如下资源：表、DL/I 数据库、MQSeries 消息和 VSAM 文件。

对于运行在 WLM-established 地址空间的存储过程而言，RRS attachment 是必需的。

2.4.3　DB2 与分布式数据

在分布式数据环境下，DB2 应用可以访问存在于不同 DB2 系统或远程的关系数据库系统中的数据。

例子：假设一家公司需要满足位于很多地方的客户请求，而且能够满足这些请求的公司客户代表工作在一个相当宽广的物理区域当中。用户可以将这些请求保存在安装有 DB2 Connect 的工作站中，这个信息将被上传到 DB2 UDB for z/OS 系统中。然后，客户代表就可以在自己的办公室中使用 Java 应用访问客户请求的信息。

这家公司的分布式环境依赖于分布式数据工具（Distributed Data Facility，DDF），DDF 也是 DB2 UDB for z/OS 的一部分。DB2 应用能够利用 DDF 访问位于其他 DB2 系统或者远程关系数据库系统中的数据，这些系统需要支持分布式关系数据库体系结构（Distributed Relational Database Architecture，DRDA）。DRDA 是一种分布式连接的标准，所有的 IBM DB2 服务器都支持这种 DRDA 标准。

对于运行在支持 DRDA 的远程环境下的应用，DDF 也能够使这些应用访问 DB2。这些应用利用 DDF 访问 DB2 服务器中的数据。IBM DB2 Connect 和其他兼容 DRDA 的客户端产品都是这种类型的应用。

使用 DDF，用户可以同时有 150 000 个分布式进程连接到 DB2 服务器。一个进程是一个 DB2 结构，它描述了一个应用的连接，并且跟踪它的进度。

使用存储过程可以降低分布式访问对于处理器和时间的消耗。一个存储过程是一个用户编写的 SQL 程序，请求者可以在服务器端调用它。通过将 SQL 封装在一起，可以减少链路的消息数量。

本地的 DB2 应用也可以使用存储过程，以利用它们封装的能力，在不同的应用之间共享 SQL。

2.4.4 DB2 与z/OS

z/OS 是继 OS/390 之后的主机操作系统。z/OS 和 IBM zSeries 800、900、990 所组成的体系结构为 e-business 提供了关键的高品质服务。z/OS 操作系统基于 64 位的 z/Architecture。这个操作系统具有高安全性、高扩展性和高性能的特点。基于这些特性，z/OS 为 Internet 和基于 Java 的应用提供了一个坚强的基础，并且为运行应用提供了全面而多样的环境。

2.4.5 DB2 与Parallel Sysplex

Parallel Sysplex 是 DB2 和 ZSeries 协同工作的一个典型范例。基于 ZSeries 的超强处理能力，DB2 利用 ZSeries Parallel Sysplex 的优势，可以建立共享组允许两个或者更多的处理

器共享同样的数据，用户可以以最小的代价获得最大化的性能，提高系统可用性和并发性，扩展系统能力，更加灵活地配置系统环境。

共享 DB2 必须隶属于一个 DB2 数据共享组。一个 DB2 数据共享组是一个或者多个 DB2 子系统的集合，这些子系统访问共享的 DB2 数据。任何一个隶属于特定数据共享组的 DB2 子系统是那个组的一个成员。一个组的所有成员使用共享的 DB2 Catalog 和 Directory。

基于数据共享，通过增加处理器和 DB2 MEMBER 给数据共享组，用户可以逐步扩展系统的处理能力。不必将负载转移到另外一个系统上，也没有必要管理数据的备份或者使用分布式处理方式访问数据。

用户可以更加灵活地配置环境。例如，可以裁减每个 z/OS 系统以满足其上用户的需求。对于在负载高峰期间发生的应用处理，可以通过唤醒休眠的 DB2 子系统来帮助处理负载。

2.4.6　DB2 与安全服务

用户可以使用 RACF（Resource Access Control Facility）或者一个相当的产品，来控制对 z/OS 系统的访问。当用户使用 TSO、IMS 或者 CICS 开启一个会话的时候，RACF 将检查 ID 信息以阻止对系统的非授权访问。

建议：使用安全服务器检查 DB2 用户的权限，并且保护 DB2 的资源。对 DB2 对象的大多数授权都可以被安全服务器直接控制。一个出口函数（一个 DB2 系统的扩展程序）可以集中控制对 DB2 对象的访问。

2.4.7　DB2 与DFSMS

DFSMSdfp（TM）存储管理系统（Storage Management Subsystem，SMS）可以用来管理 DB2 磁盘数据集。SMS 的目标是：通过集中控制、自动化任务和为系统管理员提供交互式控制，尽可能地自动化管理物理存储。SMS 可以减少用户对物理细节的关注，例如性能、空间和设备管理。

对性能比较敏感的数据来说，放置数据集的时候需要更多的手工控制。

数据集大于 4GB 的表空间和索引空间需要 SMS 对其进行管理。

2.4.8　DB2 与WLM

Workload Manager（WLM）是 OS/390 和 z/OS 的一个组件。WLM 为操作系统实现负

载管理的功能。负载管理的目标是均衡可用的系统资源，满足来自各子系统的工作需求。

DB2 使用 WLM 为存储过程分配负载请求。当存储过程请求到达 DB2 时，为了处理此请求，WLM 将决定是否需要额外的资源，例如一个新的 WLM 管理的地址空间。WLM 对运行在每个地址空间中的 TCB 的数量进行管理，并且在需要时启动额外的地址空间。从 DB2 for z/OS 版本 8 开始，所有新的存储过程一定要运行在 WLM 管理的地址空间中。

下面将详细介绍一下 DB2 存储过程的运行原理，如图 2.14 所示。

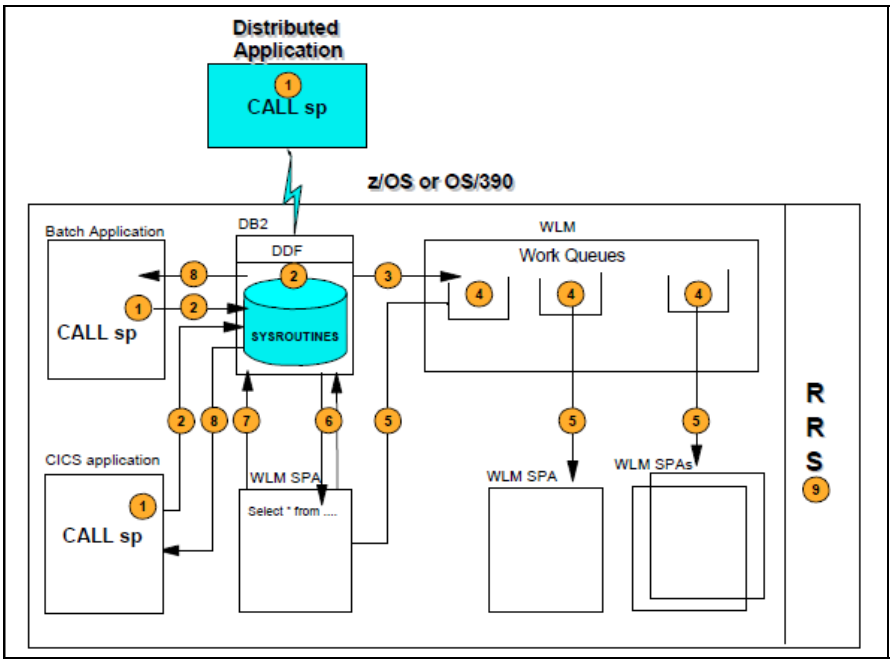


图 2.14　DB2 存储过程运行原理

当调用一个存储过程时，发生了如下事情：

（1）应用（大部分来自于客户端）执行 SQL CALL 语句调用一个存储过程。

（2）DDF 地址空间接收并处理这个请求，然后传给 DB2。在对权限和参数定义完成验证后，DB2 使用存储过程的名字在表 SYSIBM.SYSROUTINES 中进行搜寻，获取到与这个存储过程相关的 COLLID 和 WLM_ENVIRONMENT。

（3）DB2 使用下面的顺序决定 COLLID：

- 程序中的特殊寄存器 CURRENT PACKAGE PATH。
- 程序中的 CURRENT PACKAGESET。

- CREATE PROCEDURE 中的 COLLID。
- 调用程序中的 CURRENT PACKAGESET。
- 调用程序中的 COLLID。
- 调用程序中的 PKLIST。

（4）DB2 发送一个请求给 WLM，让 WLM 在应用环境中调度存储过程。

（5）WLM 将这个请求放在它的队列中。WLM 为应用环境和服务级别（SERVICE CLASS）的每个组合都维护一个队列。队列中的请求以 FIFO 的方式被处理。

（6）一旦轮到执行请求，WLM 将检查 WLM SPAS 的可用性。

（7）只要存储过程被调度执行，它就在内部执行 SQL。

（8）当存储过程到达 RETURN 语句时，它将控制返回给 DB2。依赖于程序的逻辑，可能带有返回结果。

（9）DB2 将控制返回给调用程序。依赖于程序逻辑，可能带有返回结果。

（10）因为包含存储过程的交易可跨越多个地址空间，RRS 将对交易中涉及的所有地址空间中的所有资源进行协调。

存储过程的每次执行将重复步骤 1 到步骤 8。如果一个存储过程调用了另外一个存储过程，那么上面的所有步骤将再次执行。



2.5　DB2 并发控制机制

并发控制机制是衡量一个数据库系统性能的重要指标之一。数据库系统的并发控制协调并发操作以保证事务的隔离性，保证数据的一致性。DB2 以事务为单位，通过使用锁来实现并发控制。本节将介绍 DB2 并发控制的基本原理，对数据对象施行封锁的方法，以及如何检查和管理锁。

2.5.1　数据一致性

DB2 作为多用户数据库系统，可同时运行多个事务并行存取数据，充分利用数据库资源，发挥数据库共享资源的特点。在这种情况下，可能出现多个并发的事务同时存取同一份数据的情况。在并发操作中，需要保证多个用户程序执行时数据的一致性、完整性。什么是数据一致性？回答这个问题通过观察一个实例可能是最好的方法。

例如，账户表的余额是 1000 元，当两个事务 T1 和 T2 同时更新余额表记录时，他们读取数据并修改，事务 T1 提交的结果是修改为 900 元，但在 T1 处理数据的同时，事务 T2 修改余额为 500 元，在 T2 不知道 T1 已进行了修改的情况下，账户余额变成了 500 元，丢失了 T1 事务对数据的更新，账户表的记录不能反映真实的情况，数据变得不一致。

在并发事务处理过程中，还有可能出现事务 T1 修改某一条记录，将其写入数据库，事务 T2 读取同一条记录后，T1 由于某种原因被撤销，此时 T1 已修改的数据恢复原值，T2 读取的数据就与数据库中的数据不一致，该数据就为“脏”数据。

在上面的示例中，使用了并发事务模拟共享系统中的数据处理情况，可以看出，尽管事务本身执行是正确的，但是由于相互间的干扰，可能会产生错误的总体结果。还有可能用户在执行过程中数据库系统出现异常故障导致数据库中的数据出现不一致现象。

为了防止出现数据不一致现象（特别是在多用户环境中），DB2 设计者引入了下面的数据一致性支持机制：

- 事务
- 锁定
- 日志

2.5.2 DB2 事务的概念

事务（Unit Of Work，简称 UOW，工作单元）又称为交易，由一系列的应用程序和数据库之间的互动组成一个访问数据库系统的可恢复的最小单元。事务处理是数据库的主要工作，事务是数据库应用程序的基本逻辑单元。DB2 采用事务来确保数据的完整性。如果某一事务完成，则该事务进行的所有数据更改均会提交，成为数据库中的永久组成部分。如果事务遇到错误且必须取消或回滚，则所有数据更改均被清除。

事务在可执行的 SQL 语句第一次执行时会自动初始化，事务一旦初始化，就必须由用户或启动它的应用程序来显式终止。多数情况下，事务是通过执行 COMMIT 或 ROLLBACK 语句终止的。当执行 COMMIT 语句后，在事务执行初始化之后对数据库做出的所有改变都将成为永久的（已落实）。当执行 ROLLBACK 语句后，在事务执行初始化之后对数据库做出的所有改变都会恢复，数据库返回（回滚）到它在事务开始之前的状态。不管哪种状态，数据库在事务完成时都保证能够回到一致状态。需要注意的是，虽然对数据的修改在事务被成功 COMMIT 之后才会变成永久的，这使事务提供了一般的

数据一致性，但是否能确保每个事务中 SQL 操作的顺序总会使数据库保持一致还要取决于用户或应用程序。

2.5.3　COMMIT和ROLLBACK操作的结果

前面提到，在多数情况下事务是通过执行 COMMIT 或 ROLLBACK 语句中止的，执行 COMMIT 语句后，在事务初始化后对数据库做出的所有改变都会变成永久的；在执行 ROLLBACK 语句后，在事务初始化后对数据库做出的所有改变都会被撤销，数据库则返回到它在事务开始之前的状态。为了理解这些语句都是如何工作的，看看下面的示例会有所帮助。

如果下面的 SQL 语句按照所示的顺序执行。

```
CONNECT TO DSNPB01
CREATE TABLE DEPARTMENT（DEPT_ID INTERGER NOT NULL,DEPT_NAME VARCHAR（20））
INSERT INTO DEPARTMENT VALUES（100,'PAYROLL'）;
INSERT INTO DEPARTMENT VALUES(200,'ACCOUNTING');
COMMIT;
INSERT INTO DEPARTMENT VALUES（300,'SALES');
ROLLBACK;
INSERT INTO DEPARTMENT VALUES（500,'MARKETING'）;
COMMIT;
```

一个名为 DEPARTMENT 的表被创建，如下所示：

```
DEPT_ID             DEPT_NAME
  100                 PAYROLL
  ➢ ACCOUNTING
  500                MARKETING
```

这是因为，在第 1 个 COMMIT 语句执行后，创建名为 DEPARTMENT 的表及向 DEPARTMENT 表插入两条记录这两个操作都会变成永久的。反之，在 ROLLBACK 语句执行后，向 DEPARTMENT 表插入的第 3 条记录将被删除，表会返回到它在插入操作执行之前的状态。最后，在第 2 个 COMMIT 语句执行后，向 DEPARTMENT 表插入第 4 条记录的操作将变成永久的，数据库再次返回到一致状态。

2.5.4　不成功的事务的结果

上述示例展示了当事务被 COMMIT 或 ROLLBACK 语句中止后会发生什么，但是如果

事务完成之前系统发生故障又会怎么样呢？在这种情况下，DB2 数据库管理器（DB2 Database Manager）将撤销所有未 COMMIT 的修改，从而恢复它假定在事务初始化后存在的数据库一致性。这是通过使用日志文件实现的，日志文件包含关于事务执行的每个 SQL 语句的信息，以及事务是否被成功 COMMIT 或 ROLLBACK 的信息。

2.5.5 事务隔离级别

维护数据库一致性和数据完整性，同时允许多个应用程序同时访问相同数据，就叫做并行性。DB2 系统试图实现并行性的方法之一就是使用隔离级别，这种方法将决定如何在访问的事务中使用数据是锁定该数据或将其与其他事务隔离。

事务的隔离级别保证一个事务的执行不受其他事务的干扰。设置事务隔离级别是对事务执行过程中所有访问数据的操作指定默认的加锁方式。基于设置的事务隔离级别，DB2 会判断授予各种级别的锁。

设置事务隔离级别虽然可能带来某些完整性方面的风险，但可以换取更高的并发访问的能力。每个隔离级别度提供了更大的隔离性，这是通过在更长的时间内使用更多的限制锁换来的。事务隔离级别的高低和事务的并发能力成反比。

2.5.5.1 事务隔离级别的种类

ISOLATION 级别由高到低共分为四级：CS、RR、RS、UR。

（1）ISOLATION（CS）：CS 是 Cursor Stability 的缩写，或称为游标稳定性。

当 CS 在行上定位游标时会锁定任何由应用程序的事务所存取的行。此锁定在读取下一行或中止事务之前有效。然而，改变了某一行上的任何数据，则在对数据库更改之前必须持有该锁定。

对于具有 CS 的应用程序已检索的行，当该行上有任何可更新的游标时，任何其他应用程序都不能更新或删除改行，游标稳定性应用程序不能查看其他应用程序的未 COMMIT 的更改。CS 提供了最大级别的并发行。但是在同一个程序单元，如果同一个游标被处理两次，可能返回不同的结果。扫描一个 10 000 行的表，如果采用 CS 方式，将之锁定当前游标位置之下的行，当游标移离该行时，也就除去了该锁定（除非更新该行）。使用游标稳定性是默认的隔离级别，且应在需要最大并行性，但只看到其他应用程序中已 COMMIT 的情况下才使用。

（2）ISOLATION（RR）：RR 是 Repeatable Read 的缩写，或称为可重复读。RR 会锁定

应用程序在工作单元中引用的所有行，直到 COMMIT。其他程序不能修改该数据，在同一个程序单元，如果数据被访问两次，返回相同的结果。利用 RR，不会出现丢失更新等现象，但是由于 RR 锁定的不仅仅是检索的那一行数据，而是工作单元引用的每一行，这将意味着，如果用户扫描 10 000 行数据并对它进行过滤，尽管只有 10 行满足条件，但仍然会锁定 10 000 行数据，其他工作单元不能访问被该工作单元引用的数据，并发度大大降低。同时，采用 RR 会获得和持有大量锁，有可能超过系统定义的持有锁数量的限制。

（3）ISOLATION（RS）：RS 是 Read Stability 的缩写，或称为读稳定性。RS 只会锁定应用程序在工作单元中检索的那些行，但是其他程序可以对其执行 INSERT 操作。在同一个程序单元，如果数据被访问两次，那么可能返回新插入的数据，但返回的旧数据不会变化。与 RR 不同的是，RS 只锁定限定的行。在 RS 级别，只检索 10 行，且只对那 10 行持有锁定，而不是像 RR 那样锁定 10 000 行，因此锁定的数量大大减少，并发度得到提升。适合在并发环境下运行，在工作单元中不会多次发出相同的查询，或者在同一个工作单元中发出多次查询并不要求该查询获得相同的结果的应用程序。

（4）ISOLATION（UR）：UR 是 Uncommitted Read 的缩写，或称为未提交读，也就是所谓的“脏读”。UR 不会加任何的锁，可以读数据库中任何的数据（包含已经修改但尚未 COMMIT 的数据），读的数据可能与真实的数据有一定差距。再次引用扫描 10 000 行的示例，如果使用 UR，则不需要对任何行锁定。UR 级别最常用于只读表上的查询，或者若只执行查询语句且不关心是否可从其他应用程序中看到未 COMMIT 的数据时常用。

无论选择的 ISOLATION 级别是什么，当程序执行 COMMIT 后，会自动释放所有的锁。

2.5.5.2　隔离级别的选择

为一个应用程序选择适当的隔离级别，对于避免该应用程序产生无法容忍的现象很重要。因为获取和释放锁定所需的 CPU 和内存资源随隔离级别的不同而不同，所以此隔离级别不但影响应用程序之间的隔离程度，而且还影响个别应用程序的性能表现。潜在的死锁情况随隔离级别的不同而不同。

一个基本的原则如表 2.2 所示。

表 2.2　选择隔离级别的准则

应用程序类型	需要高数据稳定性	不需要高数据稳定性
读写事务	RS	CS
只读事务	RR 或 RS	UR

2.5.6 锁机制

锁（LOCKING）是最常采用的并发控制机制。锁是事务对某个数据库中的资源（如表和记录）存取前，先向系统发出请求，封锁该资源。事务获得锁后，即获得对数据的控制权，在事务释放它的锁之前，其他事务不能更新此数据。当事务结束或撤销之后，释放被锁定的资源。

2.5.6.1 锁的大小

锁的大小，也称为粒度。在一个对象上的锁的大小指的是锁控制的数据总量。DB2 支持 5 个级别的锁：Tablespace LOCK、Table LOCK、Partition LOCK、Page LOCK 和 Row LOCK。

图 2.15 展示了 DB2 锁的层次结构。

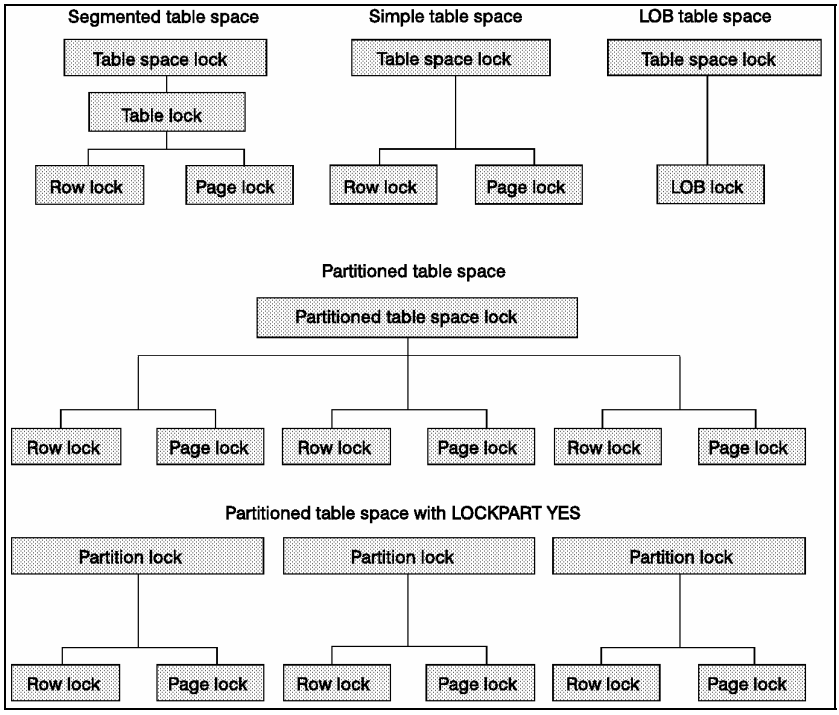


图 2.15　DB2 锁的层次结构

在锁的层次结构中，可以在任何级别上加锁。如果在上一级别上加锁，会自动继承到下一级别。例如如果用户拥有 Tablespace 级别的锁，就无须再申请 Page 锁或 Row 锁。另外如果用户在较高级别上没有一个兼容的锁，那么就不能在较低级别上加锁。

从图 2.15 中可以看出，Tablespace 锁可以控制整个 Tablespace 中所有的数据，也是最大的锁。Page 锁或 Row 锁只能控制数据的一页或一行。Page 锁和 Row 锁是在同一层次上的，即一个表或表空间不能同时拥有 Page 锁和 Row 锁。

对于不同类型的表空间类型，锁的层次略有不同，对于分区表空间，可以通过指定 LOCKPART 参数来控制分区表锁控制的数据量，设置为 LOCKPART=YES，对于分区表，如果申请 Tablespace 锁，其实 DB2 授予的是 Partition 锁，仅生效在对应分区上，而不是整个 Tablespace；设置为 LOCKPART=NO，对于分区表，如果申请 Tablespace 锁，则锁住整个 Tablespace。

2.5.6.2　锁的类型（Mode）

在 DB2 系统中，基本锁的类型有两种：排它锁（Exclusive Lock）和共享锁（Share Lock）。

1. 排它锁（X 锁）

排它锁又称为“写锁”，如果事务 T 对数据对象 R 加上 X 锁，则只允许 T 读取和修改 R，其他的任何事务都不能再对 R 加任何类型的锁，直到 T 释放 R 上的 X 锁。这就保证了在 T 释放对 R 的封锁之前，其他事务都不能再读取和修改 R。

使用 X 锁，可以避免“丢失更新”。在一个并发的联机系统中，如果事务 T1 在修改 R 前对 R 加 X 锁，则事务 T2 想要更新 R 的数据必须等待，直到 T1 完成更新后释放 X 锁，T2 才能对 R 加 X 锁。这时 T2 读到的 R 是被 T1 更新后的值，避免了丢失更新。

使用 X 锁，还可以避免读“脏”数据。如前例，如果事务 T1 在更新 R 的记录前对该表加 X 锁，那么另一个事务 T2 查询 R 表的记录也将一直处于等待状态，直到事务 T1 结束或者撤销，释放 X 锁，T2 才能够检索 R 表的记里，避免了读“脏”数据。

2. 共享锁（S 锁）

共享锁又称为“读锁”，如果事务 T 对数据对象 R 加 S 锁，则其他事务也可以对 R 加 S 锁，但不能对 R 加 X 锁，直到 R 上的所有 S 锁释放为止。即共享锁能够阻止对已加锁的数据的更新操作。

对事务中操作的数据对象加 X 锁可以避免数据一致性被破坏，但是却降低了并发性。使用共享锁，主要防止在读操作期间对数据的修改，因此允许其他事务对数据对象加 S 锁，完成对同一数据对象的并行读取操作。

如果事务 T1 在检索 R 表之前对该表加共享锁，那么在 T1 结束前，T2 不能够获得 R 表的排它锁，即不能插入新的记录。在 T1 完成本次操作中对 R 表的检索后，事务 T1 结束，

这时 T2 事务才能够修改 R 表。

2.5.6.3 锁的兼容性

锁的兼容性是指在一给定状态下，如果另一个进程持有或正在请求对同一个资源的锁定，那么是否准许锁定请求。“否”指示请求者必须等待，直到所有不兼容的锁定被其他进程释放为止。注意，当等待锁定时，可能出现超时。“是”指示准许该锁定（除非别的人正在等待该锁定）。

	S	X
S	是	否
X	否	否

假定程序 1 持有对某个表的锁定，而程序 2 也想要存取该表，数据库管理程序代表程序 2，请求某特定方式的锁定。如果由程序 1 持有的锁定方式许可程序 2 请求的锁定，则这两个锁定（或方式）被称为是兼容的。如果程序 2 所请求的锁定方式与程序 1 所持有的锁定不兼容，那么程序 2 就不能继续运行。相反，它要等到程序 1 释放其锁定，并且所有其他现存不兼容的锁定都被释放为止。图 2.16 就演示了这一过程。

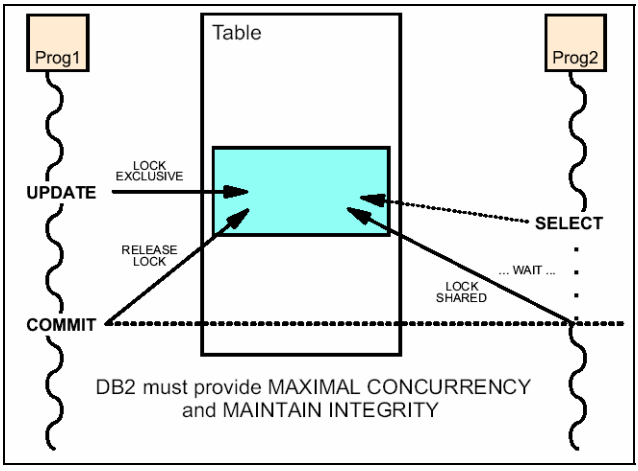


图 2.16 锁的兼容性示例

2.5.6.4 锁的持续时间

锁定的持续时间基于为请求锁定的程序所选择的 BIND 参数。当计划被请求执行时，用户可以马上获得锁定，或在程序的执行期间反复地锁定。当计划被终止或当工作单元不

再需要锁定时，锁定可以被释放。

2.5.6.5　锁定逐步升级

锁定逐步升级是减少持有的锁定数的一种内部机制。升级是从多行锁定（单个表中的）至单个表的锁定。如果当前持有太多锁定（任何类型），则将进行锁定逐步升级。如果一个特定的应用程序超过为它分配的锁定列表，可对该程序进行锁定逐步升级。

这个逐步升级在 DB2 内部处理，唯一在外部可以检测的结果可能是对一个或多个表并行存取的减少。在正常情况下，在一个配置合适的数据库系统中，不建议进行频繁的锁升级。由于锁升级后导致潜在的并发度的下降，在多用户环境下会产生性能隐患，用户也可以通过参数控制是否启用锁升级的机制。

2.5.7　锁的挂起、超时和死锁

2.5.7.1　挂起

一个事务拥有对某一资源锁的时间越长，对其他事务的潜在影响就越大。当一个应用请求一个已经被其他进程占有的锁，而且该锁不能被共享时，该应用就被挂起。一个挂起的进程临时停止运行，直到该锁定可以被获得。

2.5.7.2　超时机制

当应用已经被挂起了一段预先确定的时间后，它将被自动终止。当一个进程因为超过了这段时间而被终止时，就是所谓的“超时”。“超时”是由于一个给定资源的不可用引起的。DB2 通过设置超时机制，避免应用长时间挂起。DB2 的超时时间由 DSNZPARM 中的 IRLMRWT 参数设置，一般是 30 秒。

2.5.7.3　死锁

死锁是一种可能发生在任何多线程系统中的状态，指在同处于等待状态的两个或多个事务中，其中的每一个事务在它能够执行之前，都等待着某个数据，而这个数据已被其他的事务锁定，这种状态成为死锁。

例如，事务 T1 具有 Supplier 表上的 X 锁，之后要求对 Part 表进行锁定，事务 T2 具有 Part 表上的 X 锁，并且之后需要申请对 Supplier 表的锁。事务 T1 需要在 Part 表上加锁，但 T1 无法获得，因为 T2 已经将它锁定了，T2 也无法获得 Supplier 表的锁，因为 T1 已经

拥有它。T1、T2 相互等待对方释放已锁定的资源，因为事务需要获得已被对方锁定的资源才能进行后续的操作，而事务在 COMMIT 或 ROLLBACK 之前不能释放自己持有的锁，这种情况下 T1、T2 既不能 COMMIT 也不能 ROLLBACK，从而形成死锁。

当死锁发生后会大大降低系统的效率。当几个需要长时间运行的事务在同一个数据库中并发执行时，可能会发生死锁。死锁还可能因为数据处理的顺序原因而发生，所以解决死锁的办法有两个：一种是避免死锁的发生；另一种是设置超时机制，当死锁发生时解决死锁。

2.5.7.4 解决死锁的方法

1. 按照同一顺序访问资源

如果所有并发事务按预先规定的一个顺序访问资源，发生死锁的可能就会降低。但是在数据库系统中，并发处理的事务非常多，锁定的对象范围也很广，而且随着数据的插入、更新等操作的不断变化，要维持这样顺序的资源锁定顺序非常困难，只能在并发程度非常高的特定的应用设计中尽量按照这一原则。

2. 避免事务中的用户交互

在应用设计中尽量避免编写包含与用户交互的事务，因为运行没有用户交户的处理速度要远远快于用户手动响应查询的速度，这将降低系统的并发性。

3. 保持事务简短或在过程中增加 COMMIT 的次数

在同一数据库中并发执行多个需要长时间运行的事务经常会发生死锁，事务运行的时间越长，其持有 X 锁的时间也越长，从而堵塞了其他获得并可能导致死锁。因此在应用设计中尽量将每个事务的颗粒度减小，如果不能减少，则在过程中增加 COMMIT 的次数，以及时释放所持有的锁。

4. 使用低隔离级别

确定事务是否能在更低的隔离级别上运行。通过设置相对较低的隔离级别，可以缩短持有 S 锁的时间，从而减少锁的争抢。

5. 设置超时机制

通过设置锁定超时周期，限制锁定资源的时间，当事务等待锁的时间到达超时期限后自动 ROLLBACK 事务，释放它所持有的所有资源。

6. 结束死锁

在 DB2 系统中，死锁检测由一个称为“锁定监视器”的线程定期执行，当 DB2 识别

死锁后，通常选择开始较晚或已做更新最少的事务作为“死锁牺牲品”来结束死锁，DB2 将作为牺牲品的事务进行 ROLLBACK，保证另一个等待的事务可以继续执行。

2.6　DB2 应用程序管理

通过调用应用程序实现对 DB2 的访问是使用 DB2 的一种非常重要的方式。但是 DB2 系统中程序的概念是怎样的？这些程序如何访问 DB2 系统中的数据呢？本节将给出这些问题的解答，介绍程序管理的一些基本概念，DB2 系统如何管理应用程序，执行 DB2 程序所需要的准备步骤等内容。另外，本节还将对存储过程、UDF、TRIGGER 等概念进行简单的介绍。通过阅读本节，读者将能够了解程序管理的基本知识。

2.6.1　基本概念

如果 DB2 应用程序包含 SQL 语句，用户就需要使用 DB2 预编译器或者一个编译器提供的 SQL 语句协处理器，对这些 SQL 语句进行处理。任何一种形式的 SQL 语句处理器都做下面的事情：

- 取代源程序中的 SQL 语句为 DB2 语言接口模块的调用语句。
- 创建一个数据库请求模块（DBRM-Database Request Module），在绑定期间，这个模块将 SQL 请求发送到 DB2 系统进行一些通信。

DBRM 必须被绑定到一个 PLAN 或者 PACKAGE，包含 SQL 语句的应用程序才能够运行。下面先介绍几个应用程序密切相关的概念。

1. DBRM（database request module）

DB2 预编译程序创建的，包含了应用程序中 SQL 语句的信息，用于 BIND 处理。

2. PACKAGE

一个包含了一组被静态捆绑的 SQL 语句的 object。

3. PACKAGE LIST

一个按一定规则命名的 package 集合，可扩充为一个 plan。

4. PLAN

在 BIND 处理中产生的控制结构，是 DBRM 和 PACKAGE 的综合，应用程序执行时被调用。

5. BIND

是将 DB2 预编译程序的输出转变为可控制结构（一个 package 或 application plan）的一种处理方式，通过这种处理选择存取数据的路径和执行某种权限的核对。

6. REBIND

当某种改动会影响 PACKAGE 但不会改变程序中的 SQL 语句时，则需要对这个 APPLICATION PACKAGE 做一次 REBIND。

7. Consistency token

Consistency token 是一个时间信息，常用于确定一个应用的版本标识符。当预编译一个含有 SQL 语句的程序时，预编译程序根据 consistency token 确定对 DB2 的每一次 call。由预编译产生的 DBRM 和与之捆绑在一起的 PLAN 或 PACKAGE 根据相同的 token 匹配起来。

2.6.2 应用程序的准备过程

图 2.17 说明了应用程序的准备过程。

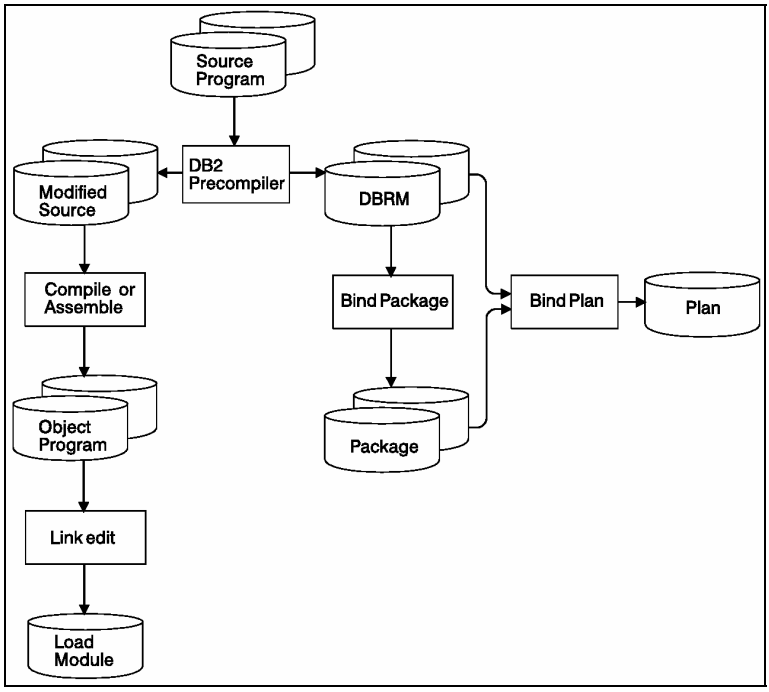


图 2.17 应用程序的准备过程

对这些步骤的说明如下。

【预编译】

在编译或者汇编传统语言程序之前，必须对嵌入在程序中的 SQL 语句进行预处理。DB2 预编译器可以对 C、COBOL、Fortran、PL/I 和汇编语言程序的 SQL 语句进行预处理。因为 SQL 语句不被大多数编译器所识别，所以为了预防编译错误，在对程序进行编译之前，必须使用 DB2 预编译器对程序进行处理。预编译器对程序进行扫描，返回修改过的源代码，然后方可以对这个源代码进行编译和连接编辑。

另外，在编译 C、C++、COBOL 和 PL/I 程序时，也可以使用 DB2 协同处理器。在编译的同时，DB2 协同处理器实现预编译的功能。

预编译器的主要输出是数据库请求模块（Database Request Module，DBRM）。一个 DBRM 是一个包含 SQL 语句和宿主变量信息的数据集，这些信息都是在程序预编译期间从源程序中解析出来的。DBRM 的目的是通过绑定操作，让 SQL 请求和 DB2 之间进行通信。

【绑定】

用户必须使用 BIND 命令将 DBRM 绑定到一个 PLAN 或者 PACKAGE，DB2 应用程序才可以运行。例如，为了将一组 SQL 语句预编译到相同的 DBRM 中，可能需要将这些 SQL 语句都放到同样的程序中，然后将它们绑定到一个单一的 PACKAGE。当程序运行的时候，DB2 使用一个时间戳来验证这个程序是否匹配了正确的 PLAN 或者 PACKAGE。

一个 PLAN 可能包含 DBRM 或 PACKAGE 列表（PACKAGE 的集合），或者二者都有。PLAN 一定包含至少一个 PACKAGE 或者一个直接绑定的 DBRM。每个 PACKAGE 可以包含且仅可以包含一个 DBRM。

一个 COLLECTION 是一组相关的 PACKAGE。绑定 PACKAGE 到 PACKAGE COLLECTION 允许用户增加 PACKAGE 到一个已经存在的应用 PLAN，而无须再次绑定整个 PLAN。当用户 BIND 一个 PLAN 的时候，如果在 PACKAGE 列表中包含一个 COLLECTION，那么 COLLECTION 中的任何 PACKAGE 对 PLAN 都是可用的。当第一次绑定 PLAN 的时候，这个 COLLECTION 甚至可以是空的。之后，无须再次绑定 PLAN，就可以逐渐增加 PACKAGE 到这个 COLLECTION，并且可以删除或者替换已经存在的 PACKAGE。

特殊寄存器——CURRENT PACKAGE PATH——说明了一个值：当对引用的 PACKAGE 进行解析的时候，这个值指出了 DB2 可以使用的 COLLECTION 列表。

【编译，连接编辑】

为了使应用程序能够与 DB2 系统进行交互，必须使用连接编辑（link-edit）过程来建立一个可执行的 LOAD MODULE 来满足环境（如 CICS、IMS、TSO 或者 batch）的要求。LOAD MODULE 是一个装载入内存中的可执行程序单位。

【运行】

在完成上面的步骤之后，就可以运行 DB2 应用程序了。为了使应用程序运行，有许多方法可以使用，如下：

- 使用 DB2 交互面板（DB2I）。它将指导你一步步完成从预先处理程序到运行程序的所有步骤。
- 在 TSO 前台或后台以批量的方式提交一个应用。
- 在 TSO 前台或者以批量的方式启动一个程序预先处理命令列表（CLIST）。
- 使用 DSN 命令解释器。
- 在 DB2 安装时，使用包含在系统数据集（例如 SYS1.PROCLIB）中的 JCL 过程。

2.6.3 存储过程

存储过程也是一个用户编写的应用程序，在另外一个程序中可以使用 CALL 语句调用存储过程。存储过程是一个编译的程序，这个程序存储在 DB2 服务中，并且可以执行 SQL 语句。

存储过程既可以在本地调用，也可以在远程调用。尤其在分布式环境下，存储过程是非常有用的，因为它们能够提高分布式应用的性能，例如：

（1）减少网络上的信息通信量。

（2）为调用远程程序提供了一种简单的方法。

当与标准的分布式应用（如图 2.18 所示）比较时，存储过程的优势是非常明显的。从 2.18 图中我们看到：对于标准的分布式应用，对于每一条 SQL 语句，客户都要与服务器进行通信。

对于每一条嵌入在程序中的 SQL 语句，客户端都要通过网络，使用发送和接收操作与服务器端进行通信。结果就是：因为网络传输时间的原因，程序耗时增加了，拿锁的时间更长，直到 COMMIT，DB2 才会释放锁。

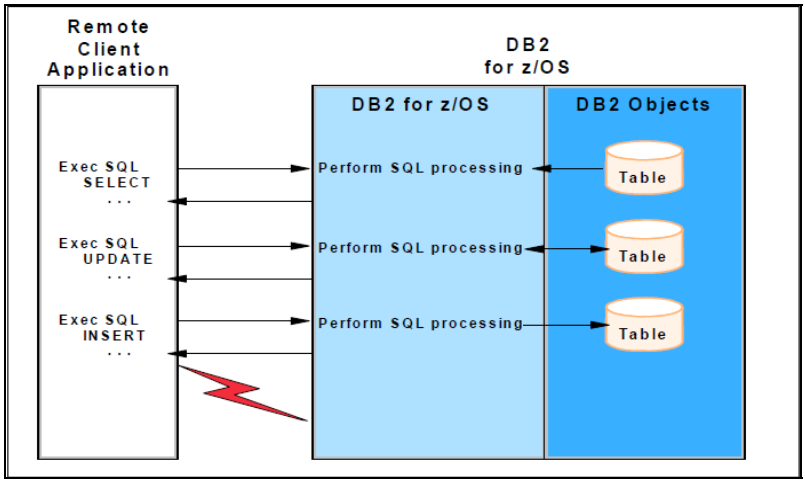


图 2.18　标准的分布式应用

图 2.19 展示了使用存储过程的情况。我们将嵌入 SQL 移到服务中，这样就能够将网络传输缩减为一条单一的 CALL 和 RETURN。

之前客户端执行的 SQL 已经被存储到服务器中，并在需要的时候由客户调用。这种调用被作为一个常规的外部调用，例如：

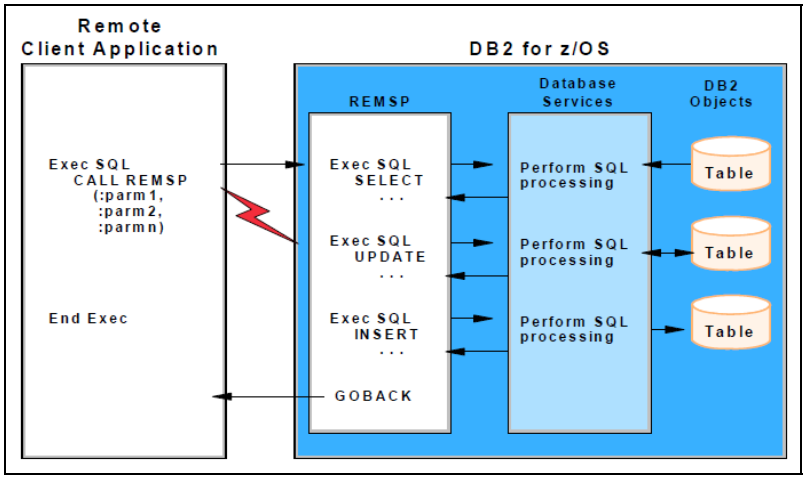


图 2.19　DB2 存储过程示例

（1）应用程序等待存储过程返回。

（2）存储过程可以传入、传出参数。

对于开发健壮的分布式应用来说，存储过程是数据库的一个关键特点。它们能够以数据库服务器支持的任何语言来编写，如 REXX、COBOL、C 等。

2.6.3.1 存储过程的类型

依据实际编码方式的不同，有两种类型的存储过程：

（1）外部存储过程（External procedure）。

（2）SQL 存储过程。

对于 SQL 存储过程而言，SQL 是唯一的编码语言，并且程序逻辑也是定义的一部分。对外部存储过程而言，程序逻辑和定义是两个分开的部分。

【外部存储过程】

外部存储过程是开发人员使用服务器支持的编程语言编写的。外部存储过程完成编写后，需要对源程序进行预编译、编译、链接和绑定，以创建 LOAD MODULE 和 PACKAGE。

外部存储过程的源码与存储过程的定义是分开的。存储过程的 LOAD MODULE 一定要被放置到 LOAD 库中，而且这个 LOAD 库需要包含在 WLM 的启动作业的 STEPLIB DD 语句。

CREATE PROCEDURE 语句被用来通知系统 LOAD MODULE 的名字，和被调用的存储过程期望的参数类型，以及其他的一些执行参数和环境参数。

下面就是一个 CREATE PROCEDURE 的例子，这个语句展示了 DB2 需要定位的 LOAD MODULE 和编写存储过程的语言：

```
CREATE PROCEDURE XFEREMP
    (parameter information)
    EXTERNAL NAME XFEREMP
```

【SQL 存储过程】

SQL 存储过程是使用 SQL 语言编写的存储过程。

与外部存储过程类似，SQL 存储过程也由存储过程的定义和存储过程程序代码组成。两种存储过程的定义在说明代码的位置方面是不同的。一个外部存储过程定义时，需要说明 LOAD MODULE 名字；而一个 SQL 存储过程定义时，就已经包含了存储过程的源代码。

下面就是一个创建 SQL 存储过程的例子：

```
CREATE PROCEDURE UPDATE_SAL
( IN INRATE DECIMAL (7,2), IN INEMPNO CHAR(6))
  LANGUAGE SQL
```

```
    UPDATE EMP
      SET SALARY = SALARY * INRATE
WHERE EMPNO = INEMPNO
```

2.6.3.2　存储过程相关的DB2 catalog tables

存储过程是 DB2 对象，它们一定要使用 DDL 进行定义。存储过程定义后，相关信息将会存储在两张 catalog 表中：SYSIBM.SYSROUTINES 和 SYSIBM.SYSPARMS。

SYSIBM.SYSROUTINES 为每个建立的存储过程都包含一条记录。这张表中包含的信息来自于 CREATE PROCEDURE 语句。这张表包含的字段描述了运行时环境、语言、参数个数、参数类型和是否返回结果集等信息。

SYSIBM.SYSPARMS 为存储过程中定义的每个参数都包含一行。这些参数信息来自于 CREATE PROCEDURE 语句。这张表的字段描述了参数的定义：名称、数据类型、输入/输出。

图 2.20 展示了 SYSIBM.SYSPARMS 和 SYSIBM.SYSROUTINES 表中记录的关系。

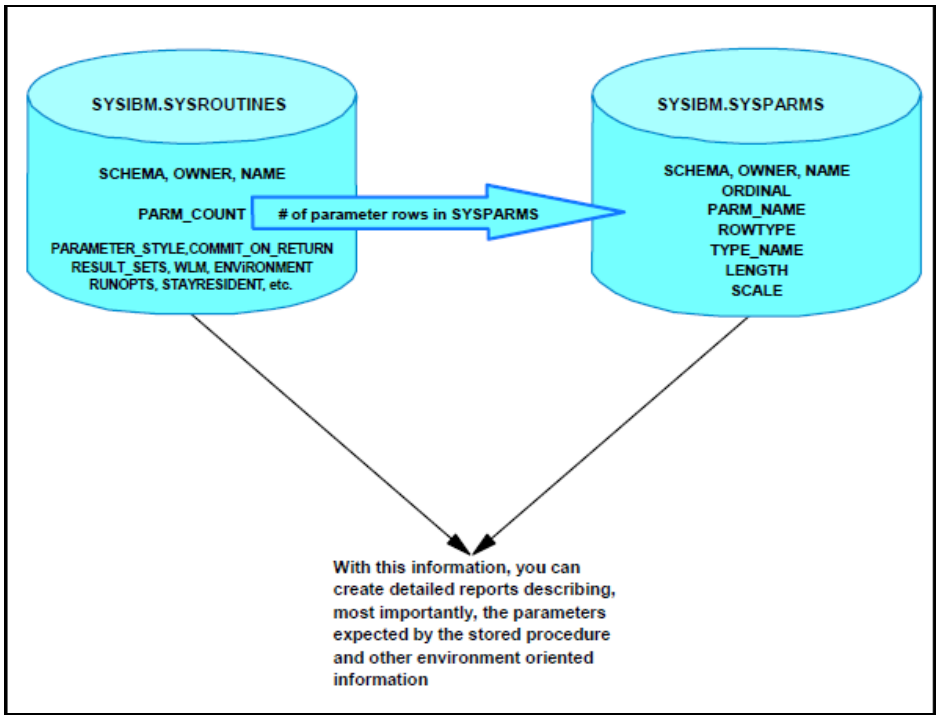


图 2.20　SYSIBM.SYSPARMS 和 SYSIBM.SYSROUTINES 记录的关系

2.6.4 UDF

UDF（User Defined Function）是用户编写的函数，作为数据库函数集合的扩充，用户可以像使用其他内置函数一样来使用它们。因为能够在 SQL 语句中调用 UDF，所以对于一个 SQL 应用来说，UDF 经常是一个更好的选择。

UDF 分为以下几种。

- Sourced UDF：基于已经存在的内部函数或者 UDF。
- External UDF：程序员使用宿主语言编写的函数。
- SQL UDF：在 UDF 的定义中包含 UDF 的源代码。

UDF 也可以分为用户定义的 scalar 函数和用户定义的 table 函数两种：

- User-defined scalar function 每次被调用时返回一个值。
- User-defined table function 每次被调用时，返回给调用 SQL 语句一个 table。

External UDF 可以是用户定义的 scalar 函数或者用户定义的 table 函数。而 Source 和 SQL UDF 则只能是用户定义的 scalar 函数。

创建和使用 UDF 包含以下步骤。

STEP 01 建立 UDF 环境。

一个 UDF 环境包括一个应用地址空间、一个 DB2 系统和一个 WLM-established 地址空间，如图 2.21 所示。

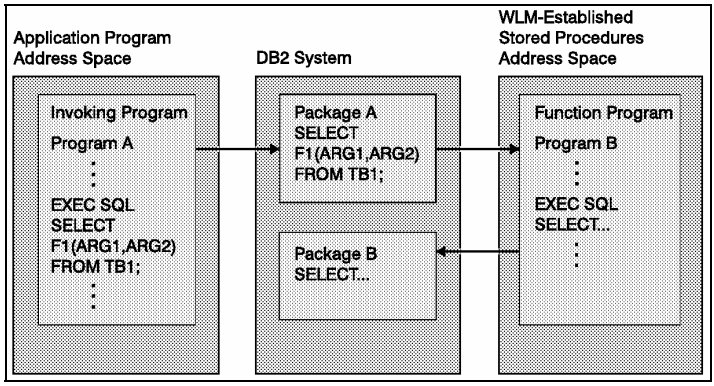


图 2.21 UDF 环境

STEP 02 编写和准备 UDF。

STEP 03 在 DB2 系统中定义 UDF。

STEP 04 从 SQL 应用中调用 UDF。

2.6.5　触发器

触发器是 SQL 语句的集合。当 DB2 表上的某个特定事件发生时，这些 SQL 语句将被执行，其中表上的特定事件可以是 INSERT、UPDATE 和 DELETE。更精确一点讲，触发器是由 DBMS（Database Management System，数据库管理系统）管理的，存储在 DBMS 之中的一组事件驱动的特殊过程。触发器不能通过直接调用来执行，它只能作为一个事件活动的结果来被 DB2 触发（执行），这个事件就是其相关的表中的数据修改动作。

触发器可以用来控制数据库中的改变。触发器能够监控很多类型的改变，并且能够实施很多操作，所以触发器的功能是非常强大的。

触发器将应用逻辑移入了 DB2 系统中，这样可以加快应用程序的开发，简化对应用程序的维护。

可以使用 CREATE TRIGGER 语句来创建触发器。在着手创建触发器之前，应当先了解下列内容：

- 使用触发器要实现的业务规则。
- 触发器是否要对其他表中的数据进行修改。
- 触发器所定义的表上还有其他触发器吗？
- 触发器将要完成的工作。
- 这些触发器创建的先后顺序。
- 新创建的触发器的触发事件（UPDATE、DELETE 或 INSERT）。
- 触发器在触发事件之前还是之后触发。

创建触发器的 DDL 语句需要下列内容。

- 触发器名（Trigger Name）：触发器的名字。
- 触发的表（Triggering Table）：触发器所定义的表。
- 触发顺序（Activation）：触发器是先于还是后于触发事件执行。
- 触发事件（Triggering Event）：导致触发器被触发的事件，它们是 INSERT、UPDATE 或 DELETE。
- 粒度（Granularity）：触发器是对每一行触发（FOR EACH ROW）还是对每一个语句

触发（FOR EACH STATEMENT）。

- 中间变量或表（Transition Variagles 或 Table）：在数据修改前或后引用这些信息的变量或表名。
- 所触发的行动（Triggered Action）：当触发器触发后执行的程序代码。

下面的例子使用 CREATE TRIGGER 语句创建一个非常简单的触发器：

```
CREATE TRIGGER SALARY_UPDATE
  BEFORE UPDATE OF SALARY
  ON DSN8610.EMP
  FOR EACH ROW MODE DB2SQL
WHEN （NEW.SALARY > （OLD.SALARY * 1.5))
  BEGIN ATOMIC
  SIGNAL SQLSTATE ‘75001’ （‘Raise exceeds 50%’）;
END;
```

这条语句创建了一个名为 SALARY_UPDATE 的 UPDATE 触发器。这个触发器将在每个受 UPDATE 影响的行中执行。如果新的 SALARY 值超过旧的 SALARY 值的 50%，它将报一个错并产生一个 SQLSTATE 码和错误信息。



2.7 DB2 应用程序 SQL 优化

如果应用程序中包含访问 DB2 系统中数据的查询，那么查询编写的好坏将在很大程度上影响应用程序的效率高低，本节将对如何编写高效的查询做出详细的说明。通过阅读本节，在编写含有 SQL 的应用程序时，对于怎样编写 SQL 才能够达到高效访问 DB2 系统的目的，读者将有一个粗略的认识。

2.7.1 应用程序编写的通用原则

如果一个查询效率很差，应首先使用下面的规则来检查一下，看看是否忽略了一些基本的东西。

2.7.1.1 编写尽可能简单的查询

确保 SQL 查询尽可能简单和高效，确保没有选择无用的 COLUMN 并且没有不需要的 ORDER BY 和 GROUP BY。

2.7.1.2　正确地编写所有的谓词

Indexable predicates：确保所有应该是 indexable 的谓词都被正确编写，保证这些谓词是 indexable 的。

Unintentionally redundant or unnecessary predicates：尽量删除那些非故意冗余的或者不需要的谓词，这些谓词会降低性能。

谓词编写的方法请参考 2.7.2 节。

2.7.1.3　查询中是否有子查询

当决定怎样或者是否使用子查询的时候，没有一个绝对的原则去遵循。下面的原则是一些通用的指导性原则：

（1）如果表上存在对子查询可用的高效索引，那么相关子查询可能是最有效率的子查询。

（2）如果表上不存在对子查询可用的高效索引，那么非相关子查询的效率可能更好。

（3）如果在一个父查询中存在多个子查询，确保以最有效率的方式排列这些子查询。

例子：假设表 MAIN_TABLE 中有 1000 条记录。

```
SELECT * FROM MAIN_TABLE
  WHERE TYPE IN （subquery 1）AND
       PARTS IN （subquery 2）;
```

假设子查询 1 和子查询 2 是相同类型的子查询（相关或者不相关），并且子查询在 stage 2，DB2 以子查询谓词出现在 where 语句中的顺序对它们进行估值。子查询 1 可以过滤掉总记录数中的 10%，子查询 2 可以过滤掉总记录数中的 80%。

子查询 1 中的谓词（称为 P1）被估值 1000 次，子查询 2 中的谓词（称为 P2）被估值 900 次，所以将进行总计 1900 次的谓词检查。然而，如果将子查询谓词的顺序颠倒，P2 将被估值 1000 次，但是 P1 将仅被估值 200 次，所以将进行总计 1200 次的谓词检查。

如果 P1 和 P2 的执行时间相同，将 P2 写在 P1 的前面好像更有效率。然而，如果估值的时候，P1 比 P2 快 100 倍，那么将子查询 1 写在前面可能更加明智。如果存在性能的下降，可以考虑调整子查询的顺序，然后监控效果。

如果还不确定，那么可以在带有相关和非相关子查询的查询上运行 EXPLAIN。通过检查 EXPLAIN 的输出，并且了解一下数据分布和 SQL 语句，就应该能够决定哪种形式更加有效。

这个通用的规则对所有类型的谓词都是适用的。然而，因为相比于其他的谓词，子查询谓词可能会消耗上千倍的 CPU 和 I/O，因此子查询谓词的顺序特别重要。

无论顺序如何，在子查询被转换成 JOIN 之前，DB2 总是先于相关子查询谓词执行非相关子查询谓词。

2.7.1.4 查询中是否包含聚集函数

如果查询中涉及聚集函数，确保尽可能简单地编写它们。这增加了数据取出时就计算它们的机会，而不是在数据取出之后才计算它们。一般来说，在数据访问期间计算这些聚集函数的效果最好，其次是在 DB2 SORT 期间进行计算。在数据取出之后才计算聚集函数是最坏的情况。

对于在数据取出时计算聚集函数的情况，查询中的所有聚集函数一定要满足下面的条件：

- GROUP BY 不需要 SORT。
- 不存在 stage 2 的谓词。
- 不存在 distinct set function，例如 COUNT（DISTINCT C1）。
- 如果查询是一个 JOIN，所有的聚集函数一定要在最后一个被 JOIN 的表上。
- 所有的聚集函数一定要在没有算术表达式的单一列上。
- 不是下面的聚集函数：
 - STDDEV
 - STDDEV_SAMP
 - VAR
 - VAR_SAMP

2.7.1.5 SQL语句的谓词中是否有输入变量

当在查询中使用宿主变量或者参数标签时，在绑定包含这个查询的包或者计划时，用户是不知道实际值的。因此，DB2 使用一个默认的过滤率，为 SQL 语句决定最好的访问路径。如果那个访问路径被证实不够高效，那么可以通过其他措施来得到一个更好的访问路径。

2.7.1.6 是否存在列相关上的问题

如果表中两列的值不能独立变化，那么这两列被称为是相关的。

当查询中包含相关列的时候，DB2 不能决定最好的访问路径。

2.7.1.7　是否可以使用非列表达式来编写查询

下面的谓词在操作符的两边放置了一个列——SALARY，和一个来自于非列的值：

```
WHERE SALARY + (:hv1 * SALARY) > 50000
```

如果以下面的方式重写谓词，DB2 能够更加高效地计算谓词：

```
WHERE SALARY > 50000/（1 + :hv1)
```

在第二种方式中，列本身在操作符的一边，所有其他的值在操作符的另一边。右侧的表达式称为非列表达式。DB2 能够在称为 stage 1 的早期处理阶段对包含非列表达式的谓词进行处理，因此查询会更快地运行完。

2.7.1.8　Materialized Query Table可能有助于提高查询的效率

在海量数据上进行操作并且包含多个 JOIN 操作的动态查询可能消耗很长的时间运行。提高这些查询效率的一种方法是：提前生成所有或者部分查询的结果，并且将结果存储在 Materialized Query Table 中。

Materialized Query Table 是用户创建的一类表，可以由用户维护或者系统维护。当 DB2 执行一个动态查询时，如果设置了参数或特殊寄存器，允许 DB2 使用 Materialized Query Table，DB2 优化器通过分析认为使用 Materialized Query Table 的内容能够获得性能上的提高，将使用 Materialized Query Table 的内容。

2.7.1.9　查询中是否包含加密数据

加密和解密能够降低某些查询的性能。然而，通过仔细地编写查询，慎重地设计含有加密数据的数据库，就可以减弱加密和解密对性能的影响。

2.7.2　编写高效的谓词

谓词能够在 SQL 语句的 WHERE、HAVING 或者 ON 字句中发现，谓词描述了数据的属性。通常情况下，它们基于表中的列；当访问表的时候返回一条记录（通过一个索引）或者过滤掉记录（通过一个扫描返回）。对于表来说，最终返回和过滤的记录独立于所选取的访问路径。

例子：下面的查询有三个谓词：C1 上的等于谓词；C2 上的 BETWEEN 谓词；C3 上的 LIKE 谓词。

```
SELECT * FROM T1
  WHERE C1 = 10 AND
       C2 BETWEEN 10 AND 20 AND
       C3 NOT LIKE 'A%';
```

因为 SQL 允许用户以不同的方式表达同样的查询，所以知道谓词如何影响访问路径选择对用户编写高效访问数据的查询及分析定位性能问题有很大帮助。

2.7.2.1 谓词的属性

选择访问路径时，HAVING 子句中的谓词不被使用，因此，本节中提到的谓词只针对 WHERE 或者 ON 子句后面的谓词。

谓词对访问路径的选择产生影响的几大因素为：

- 谓词的类型。
- 谓词是否为 indexable 的。
- 谓词是 stage 1 的还是 stage 2 的。
- 谓词中是否包含 ROWID 列。

谓词可分为如下几种：

1. Simple or compound

一个 compound 谓词是通过 AND 或者 OR 布尔操作符连接在一起的两个谓词的结果，这两个谓词可以是 simple 或者 compound 的，所有其他的谓词都是 simple 的。

2. Local or join

Local 谓词仅仅访问一张表。它们仅应用在那张表上，并且限制从那张表上返回的记录数。Join 谓词涉及多张表。它们决定了两张或者更多张表中记录的 JOIN 方式。

3. Boolean term

没有被 compound OR 谓词结构包含的任何谓词都是 Boolean term。对于一个特殊的记录，如果一个 Boolean term 计算后为 false，则整个 WHERE 子句对那条记录也为 false。

2.7.2.2 谓词类型

谓词的类型依赖于它的操作符或者语法。当计算谓词的时候，谓词的类型决定了将发生什么类型的处理和过滤。表 2.3 展示了不同的谓词类型。

表 2.3　谓词类型的定义和例子

类　型	定　义	例　子
Subquery	包含其他 SELECT 语句的谓词	C1 IN （SELECT C10 FROM TABLE1)
Equal	不是 subquery 谓词的任何其他谓词，包含相等（equal）操作符并且没有 NOT 操作符。如下形式的谓词也是 equal 谓词：C1 IS NULL；C IS NOT DISTINCT FROM	C1=100
Range	不是 subquery 谓词的任何其他谓词，并且操作符是下面列出的其中一个： >, >=, <, <=, LIKE, or BETWEEN	C1>100
In-list	具有形式的谓词： column IN （值的列表）	C1 IN （5,10,15)
NOT	不是 subquery 谓词的任何其他谓词，并且包含一个 NOT 操作符。如下形式的谓词也包括其中： C1 IS DISTINCT FROM	COL1 <> 5 or COL1 NOT BETWEEN 10 AND 20

谓词类型对访问路径影响的例子如下。

下面的两个例子展示了谓词类型如何影响 DB2 对访问路径的选择。在每个例子当中，假设在表 T1（C1，C2）上存在一个唯一索引 I1（C1），并且 C1 的所有值都是正整数。

下面的查询有一个 range 谓词：

```
SELECT C1,C2 FROM T1 WHERE C1 >=0;
```

然而，这个谓词并没有过滤掉 T1 中的任何一条记录。因此，在绑定期间，DB2 就可以决定 table space scan 要优于 index scan 的访问路径。

下面的谓词有一个 equal 谓词：

```
SELECT * FROM T1 WHERE C1=0;
```

这种情况下，DB2 会选择使用索引，因为索引在列 C1 上有很高的过滤率。

2.7.2.3　Indexable and nonindexable predicates

定义为 Indexable 谓词类型是能够匹配索引的条目；其他的类型不能。Indexable 谓词可能不会成为一个索引的匹配谓词（Matching Predicate），这取决于可用的索引和绑定时选择的访问路径。

例子：如果雇员表有一个在列 LASTNAME 上的索引，下面的谓词能够成为一个匹配谓词。

```
SELECT * FROM DSN8810.EMP WHERE LASTNAME = 'SMITH';
```

下面的谓词不可能成为一个匹配谓词，因为它不是 indexable 的：

```
SELECT * FROM DSN8810.EMP WHERE SEX <> 'F';
```

建议：为了让查询尽可能地高效，在查询中尽量使用 indexable 谓词，并且在表上创建合适的索引。Indexable 谓词允许 matching index scan，通常情况下，这都是一种非常有效的访问路径。

2.7.2.4 stage 1 and stage 2 predicates

stage1 和 stage2 分别为一个查询取出的记录需要经历两个阶段的处理：

1．stage 1 谓词（有时称为 sargable）可以应用在第一个阶段。

2．stage 2 谓词（有时称为 nonsargable 或者 residual）直到第二个阶段才可以被应用。

下面的因素决定了一个谓词是否是 stage 1 的：

- 谓词语法。
- 谓词中常量或者列的类型和长度。

从语法上被分类为 indexable 并且是 stage 1 的简单谓词，可能并不是 indexable 或者 stage 1 的，因为它包含的常量和列的长度太长了。

例子：下面的谓词不是 indexable 的：

```
CHARCOL<'ABCDEFG'，此处 CHARCOL 的定义为 CHAR（6）
```

这个谓词之所以不是 indexable 的，是因为列的长度小于常量的长度。

例子，下面的谓词不是 stage 1 的：

```
DECCOL>34.5，此处 DECCOL 的定义为 DECIMAL（18,2）
```

这个谓词不是 stage 1 的，因为 decimal 列的精度大于 15。

- DB2 在 JOIN 操作之前还是之后计算谓词。

在 JOIN 操作之后计算的谓词总是 stage 2 的谓词。

- JOIN 序列

依赖于 JOIN 序列，同样的谓词可能是 stage 1 或者 stage 2 的。JOIN 序列是计算一个查询的时候，DB2 连接表的顺序。JOIN 序列与表出现在谓词中的顺序不一定相同。

例如：下面这个谓词可能是 stage 1 或者 stage 2 的。

```
T1.C1=T2.C1+1
```

如果 T2 是 JOIN 序列中的第一张表，谓词就是 stage 1 的，但是如果 T1 是 JOIN 序列中的第一张表，谓词就是 stage 2 的。

在查询上执行 EXPLAIN 并且检查 PLAN TABLE 的结果，就可以确认 JOIN 序列。

所有的 indexable 谓词都是 stage 1 的。但是 stage1 的谓词并不一定是 Indexable 的。

建议：无论何时，尽可能地使用 stage 1 谓词。

2.7.2.5　Boolean Term(BT)predicates

一个 Boolean Term 谓词，或者 BT 谓词，是一个 simple 或 compound 谓词。对于一条特定的记录，当这个谓词计算为 false 时，整个 WHERE 子句都是 false 的。

例如，在下面的查询中，P1、P2 和 P3 都是 simple 谓词：

```
SELECT * FROM T1 WHERE P1 AND （P2 OR P3）;
```

P1 是一个 simple BT 谓词。

P2 和 P3 都是 simple non-BT 谓词。

P2 OR P3 是一个 compound BT 谓词。

P1 AND (P2 OR P3)是一个 compound BT 谓词。

BT 谓词对访问路径的影响在于：在单索引处理过程中，仅仅 BT 谓词被选为匹配谓词（Matching Predicate）。因此，仅仅 indexable BT 谓词可以作为 Matching index Scan 的候选者。为了通过非 BT 谓词匹配到索引列，DB2 考虑使用多索引访问。

在 JOIN 操作中，相比于非 BT 谓词，BT 谓词能够在更早的阶段将记录过滤掉。

建议：对 JOIN 操作来说，无论何时都要尽可能地选择 BT 谓词而不是非 BT 谓词。

2.7.2.6　ON字句中的谓词

ON 子句为外部连接（outer join）提供了连接条件。对于一个全外部连接（full outer join），ON 子句仅能够使用相等谓词（equal 谓词）。对于其他的外部连接来说，除了包含子查询的谓词，ON 子句可以使用任何一个谓词。

对于左、右外部连接和内部连接来说，ON 子句中的连接谓词被像其他的 stage 1 和 stage 2 谓词一样处理。ON 子句中的 stage 2 谓词被作为内部表的 stage 2 谓词来处理。

对于全外部连接，在 JOIN 操作期间，ON 子句像 stage 2 谓词一样被处理。

在一个外部连接中，在 JOIN 操作之后计算的谓词是 stage 2 谓词。在表表达式（Table

Expression）中的谓词可以在 JOIN 操作之前计算，因此可以成为 stage 1 谓词。

例如：在下面的语句中，谓词“EDLEVEL>100”在全连接之间计算，因此是一个 stage 1 谓词。

```
SELECT * FROM （SELECT * FROM DSN8810.EMP
WHERE EDLEVEL >100）AS X FULL JOIN DSN8810.DEPT
  ON X.WORKDEPT = DSN8810.DEPT.DEPTNO;
```

2.7.2.7 谓词计算的一般性原则

从访问路径优化的角度，对谓词计算的建议如下：

（1）从资源使用方面来说，越早计算谓词越好。

（2）因为 stage 1 谓词能够更早地过滤掉记录，并且减少 stage 2 所需的处理量，所以 stage1 谓词优于 stage 2 谓词。

（3）可能的情况下，尽量编写首先计算最严格谓词的查询。当具有高过滤率的谓词被首先处理的时候，不必要的记录能够尽可能早地被扫描处理，这样能够降低后期的处理消耗。然而，一个谓词的限制性仅仅在同样类型的谓词中和相同的计算阶段才是有效的。

2.7.2.8 谓词计算的顺序

DB2 优化器在决定维持计算的顺序时会参照以下两个规则集。

1. 第一个规则集

（1）Indexable 谓词被首先应用。在索引键值列上的所有 matching 谓词被首先应用，并且在索引被访问的时候进行计算。

接下来，没有被选为 matching 谓词，但是仍然涉及索引列的 stage 1 谓词被应用到索引。这称为索引扫描（index screening）。

（2）接下来，其他的 stage 1 谓词被应用。

在数据页面被访问之后，stage 1 谓词被应用到数据记录上。

（3）最后，stage 2 谓词将应用在返回的数据记录上。

2. 第二个规则集

描述了每个阶段谓词计算的顺序：

（1）所有的相等（equal）谓词（包括列表中仅有一个值的 column IN 列表谓词，或者 column BETWEEN value1 AND value1）被计算。

（2）所有的范围（range）谓词和 column IS NOT NULL 谓词被计算。

（3）计算所有其他的谓词类型。

在上面的两个规则集应用之后，谓词以它们出现在查询中的顺序被计算。因为顺序由用户定义，因此在计算的顺序上用户可以有一些控制。

例外情况：无论编码顺序如何，非相关子查询总是在相关子查询之前进行计算，除非 DB2 将子查询转换成 JOIN。

2.7.2.9　谓词处理的总结

编写合理高效的谓词对程序性能意义重大。本节对常用的谓词类型及处理进行了总结，供读者参考，如表 2.4 所示。

表 2.4　谓词类型及处理

谓词类型	Indexable？	Stage 1？	注　　释
COL = value	Y	Y	16
COL = noncol expr	Y	Y	9,11,12,15
COL IS NULL	Y	Y	20,21
COL op value	Y	Y	13
COL op noncol expr	Y	Y	9,11,12,13
COL BETWEEN value1 AND value2	Y	Y	13
COL BETWEEN noncol expr1 AND noncol expr2	Y	Y	9,11,12,13,23
value BETWEEN COL1 AND COL2	N	N	
COL BETWEEN COL1 AND COL2	N	N	10
COL BETWEEN expression1 AND expression2	Y	Y	6,7,11,12,13,14
COL LIKE 'pattern'	Y	Y	5
COL IN (list)	Y	Y	17,18
COL <> value	N	Y	8,11
COL <> nocol expr	N	Y	8,11
COL IS NOT NULL	Y	Y	21
COL NOT BETWEEN value1 AND value2	N	Y	

续表

谓词类型	Indexable?	Stage 1?	注　　释
COL NOT BETWEEN noncol expr1 AND noncol expr2	N	Y	
value NOT BETWEEN COL1 AND COL2	N	N	
COL NOT IN (list)	N	Y	
COL NOT LIKE ' char'	N	Y	5
COL LIKE '%char'	N	Y	1,5
COL LIKE '_char'	N	Y	1,5
COL LIKE host variable	Y	Y	2,5
T1.COL = T2 col expr	Y	Y	6,9,11,14,15
T1.COL op T2 col expr	Y	Y	6,9,11,12,13,14,15
T1.COL <> T2 col expr	N	Y	8,11
T1.COL1 = T1.COL2	N	N	3
T1.COL1 op T1.COL2	N	N	3
T1.COL1 <> T1.COL2	N	N	3
COL=(nocor subq)	Y	Y	
COL = ANY (nocor subq)	N	N	22
COL = ALL (nocor subq)	N	N	
COL op (nocor subq)	Y	Y	
COL op ANY (nocor subq)	Y	Y	22
COL op ALL (nocor subq)	Y	Y	
COL <> (nocor subq)	N	Y	
COL <> ANY (nocor subq)	N	N	22
COL <> ALL (nocor subq)	N	N	
COL IN (nocor subq)	Y	Y	24
(COL1,…,COLn) IN (nocor subq)	Y	Y	
COL NOT IN (nocor subq)	N	N	
(COL1,…,COLn) NOT IN (nocor subq)	N	N	
COL=(cor subq)	N	N	4
COL = ANY (cor subq)	N	N	22
COL = ALL (cor subq)	N	N	
COL op (cor subq)	N	N	4
COL op ANY (cor subq)	N	N	22

续表

谓词类型	Indexable?	Stage 1?	注　　释
COL op ALL (cor subq)	N	N	
COL <> (cor subq)	N	N	4
COL <> ANY (cor subq)	N	N	22
COL <> ALL (cor subq)	N	N	
COL IN (cor subq)	N	N	19
(COL1,…,COLn) IN (cor subq)	N	N	
COL NOT IN (cor subq)	N	N	
(COL1,…,COLn) NOT IN (cor subq)	N	N	
COL IS DISTINCT FROM value	N	Y	8,11
COL IS NOT DISTINCT FROM value	Y	Y	16
COL IS DISTINCT FROM noncol expr	N	Y	8,11
COL IS NOT DISTINCT FROM noncol expr	Y	Y	9,11,12,15
T1.COL1 IS DISTINCT FROM T2.COL2	N	N	3
T1.COL1 IS NOT DISTINCT FROM T1.COL2	N	N	3
T1.COL1 IS DISTINCT FROM T2 col expr	N	Y	8,11
T1.COL1 IS NOT DISTINCT FROM T2 col expr	Y	Y	6,9,11,12,14,15
COL IS DISTINCT FROM (noncor subq)	N	Y	
COL IS NOT DISTINCT FROM (noncor subq)	Y	Y	
COL IS DISTINCT FROM ANY (noncor subq)	N	N	22
COL IS NOT DISTINCT FROM ANY (noncor subq)	N	N	22
COL IS DISTINCT FROM ALL (noncor subq)	N	N	
COL IS NOT DISTINCT FROM ALL (noncor subq)	N	N	4
COL IS DISTINCT FROM (cor subq)	N	N	4
COL IS NOT DISTINCT FROM (cor subq)	N	N	22
COL IS DISTINCT FROM ANY (cor subq)	N	N	22
COL IS NOT DISTINCT FROM ANY (cor subq)	N	N	22
COL IS DISTINCT FROM ALL (cor subq)	N	N	
COL IS NOT DISTINCT FROM ALL (cor subq)	N	N	
EXISTS (subq)	N	N	19
NOT EXISTS (subq)	N	N	
expression = value	N	N	
expression <> value	N	N	
expression op value	N	N	
expression op (subq)	N	N	

表中列出了许多简单谓词，并且描述了这些谓词是否是 indexable 或者 stage 1 的。其中使用了下面的术语：

- **subq** 意味着一个相关或者非相关子查询。
- **noncor subq** 意味着一个非相关子查询。
- **cor subq** 意味着一个相关子查询。
- **OP** 代表如下操作符中的任何一个：>，>=，<，<=，¬>，¬<。
- **value** 是一个常数，宿主变量，或者特殊寄存器。
- **pattern** 是不以特殊字符（%或者_）开始的任何一个字符串。
- **char** 是不包括特殊字符（%或者_）的任何一个字符串。
- **expression** 是任何一个表达式，这个表达式中可以包括算数操作符、转换函数、聚集函数、连接函数、列、常量、宿主变量、特殊寄存器、或者日期或时间表达式。
- **noncol expr** 是一个非列表达式。这个表达式中可以包含算数操作符、转换函数、连接函数、常量、宿主变量、特殊寄存器、或者日期或时间表达式。

非列表达式的例子如下：

```
CURRENT DATE – 50 DAYS
```

- **Tn col expr** 是一个表达式，这个表达式包含表 Tn 中的一个列。这个表达式可能仅仅为那个列。
- **predicate** 是任何类型的谓词。

一般来说，如果使用 OR 操作符将几个简单的谓词结合形成一个复合谓词，那么操作的结果与最后一个计算的简单谓词拥有同样的属性。例如，如果两个 indexable 谓词被使用 OR 操作符结合到一起，结果是 indexable 的。如果一个 stage 1 谓词和一个 stage 2 谓词通过 OR 谓词结合到一起，结果是 stage 2 的。

表 2.4 的注释如下。

（1）当且仅当 ESCAPE 字符被说明，并且使用在 LIKE 谓词中时，才是 indexable 的。例如，COL LIKE ‘+%char’ ESCAPE ‘+’是 indexable 的。

（2）当且仅当宿主变量中的模式是一个 indexable 常数时，才是 indexable 的。例如，host variable=‘char%’。

（3）如果 COL1 和 COL2 都来自于同一张表，那么这两列上的索引不被考虑使用。然而，如下的查询是一个例外：

```
SELECT * FROM T1 A, T1 B WHERE A.C1 = B.C2;
```

通过使用相关的名字，这个查询将一张表当成两张独立的表对待。因此，列 C1 和 C2 上的索引被考虑用来访问表。

（4）对于一个给定的相关值，如果子查询已经被计算，那么子查询不必重新计算。

（5）如果在列上存在一个 field procedure，那么就不是 indexable 或 stage 1 的。

（6）JOIN 序列左边的列一定不同于 JOIN 序列右边的列。

（7）包含 expression1 或 expression2 中列的表一定已经被访问过。

（8）谓词 WHERE NOT COL=value 的处理与 WHERE COL<>value 类似，等等。

（9）如果 noncol expr，noncol expr1，或者 noncol expr2 是下面形式的非列表达式，那么谓词不是 indexable 的。

- noncol expr + 0
- noncol expr – 0
- noncol expr * 1
- noncol expr / 1
- noncol expr CONCAT empty string

（10）COL，COL1 和 COL2 可以是相同的列，也可以是不同的列。这些列在同一张表中。

（11）下面条件中的任何一个将使谓词成为 stage 2 的。

- JOIN 序列的左边是 DECIMAL(p,s)，此处 P>15，并且 JOIN 序列的右边是 REAL 或者 FLOAT。
- JOIN 序列的左边是 CHAR、VARCHAR、GRAPHIC 或者 VARGRAPHIC，并且 JOIN 序列的右边是 DATE、TIME 或者 TIMESTAMP。

（12）如果 JOIN 序列的左边是 CHAR 或者 VARCHAR，JOIN 序列的右边是 GRAPHIC 或者 VARGRAPHIC，并且 JOIN 序列的左边不是 Unicode mixed 的，那么谓词是 stage 1 的，但不是 indexable。

（13）如果在比较的两边都是字符串，下面的任何一个条件都将使谓词成为 stage 1，但不是 indexable 的。

- JOIN 序列的左边是 CHAR 或者 VARCHAR，JOIN 序列的右边是 GRAPHIC 或者 VARGRAPHIC。

- 下面的两个条件为真。
 - 比较的两边都是 CHAR 或者 VARCHAR。
 - JOIN 序列左边的长度小于 JOIN 序列右边的长度。
- 下面的两个条件为真。
 - 比较的两边都是 GRAPHIC 或者 VARGRAPHIC 的。
 - JOIN 序列左边的长度小于 JOIN 序列右边的长度。
- 下面的两个条件为真。
 - JOIN 序列的左边是 GRAPHIC 或者 VARGRAPHIC 的，并且 JOIN 序列的右边是 CHAR 或者 VARCHAR。
 - JOIN 序列左边的长度小于 JOIN 序列右边的长度。

（14）比较的两边都是字符串，但是两边有不同的 CCSID，当且仅当 JOIN 序列的左边是 Unicode 并且此比较不满足 13 条所列之条件时，谓词才是 stage 1 的，且是 indexable 的。

（15）在下面的情形下，谓词是 stage 2 的。

- noncol expr 是一个 CASE 表达式。
- 所有下面的条件为真。
 - noncol expr 是两个非列表达式的乘积或者商。
 - noncol expr 是一个整数值。
 - COL 是一个 FLOAT 或者 DECIMAL 列。

（16）如果 COL 拥有 ROWID 的数据类型，那么 DB2 将尽量使用 direct row access，而不是 index access 或者 table space scan。

（17）如果 COL 拥有 ROWID 的数据类型，并且有一个索引定义在这个列上，那么 DB2 将尽量使用 direct row access，而不是 index access。

（18）如果下面的条件为真，那么 IN-list 谓词就是 indexable 和 stage 1 的。

- IN 列表中仅包含一个值。例如常量、宿主变量、参数标志符或特殊寄存器。
- IN 列表中不包含任何聚集函数或者转换函数。
- IN 列表没有包含在触发器的 WHEN 字句中。
- 对于左边列是精度大于 15 的 DECIMAL 列的数字型谓词来说，IN 列表中没有 FLOAT 类型的值。
- 对字符串谓词而言，字符集符号与左边列的字符集符号相同。

- 对于 DATE，TIME，TIMESTAMP 谓词而言，左边列必须是 DATE，TIME，TIMESTAMP。

（19）COL IN (corr subq)和 EXISTS(corr subq)谓词可能成为 indexable 和 stage 1 的，如果在处理过程中，这些谓词被转换成 JOIN。

（20）当查询定义为 NOT NULL 的列时，谓词 COL IS NULL 和 COL IS NOT NULL 是 stage 2 的谓词。

（21）如果谓词类型是 COL IS NULL，并且列定义为 NOT NULL，因为 C1 不是 NULL 的，所以表不被访问。

（22）关键字 ANY 和 SOME 的功能类似。如果带有 ANY 关键字的谓词不是 indexable 和 stage 1 的，那么带有 SOME 关键字的类似谓词也不是 indexable 和 stage 1 的。

（23）在下面的情形下，谓词是 stage 2 的：

- noncol expr 是一个 CASE 表达式。
- noncol expr 两个非列表达式的乘积或商，而乘积或商是一个整数，并且 COL 是一个 FLOAT 或者 DECIMAL 列。

（24）COL IN (noncor subq)仅对 type N access 是 stage 1 的。其他情况下，这个谓词都是 stage 2 的。

2.7.2.10　谓词属性的例子

表 2.4 对谓词类型进行了总结，本节将列举一些常见的谓词例子来展示表 2.4 中的一般性谓词原则，以帮助读者对谓词有进一步感性的认识。在下面的示例中关键的假设有以下几点。

假设 1：谓词 P1 和 P2 都是简单的、stage 1 的、indexable 的谓词。

- P1 AND P2 是一个复合的、stage 1 的、indexable 的谓词。
- P1 OR P2 是一个复合的、stage 1 的谓词，但不是 indexable 的谓词，除非通过将来自于两个索引的 RID 列表进行 union 操作。

假设 2：在表中的列（C1，C2，C3，C4）上存在一个索引，并且每个列中 0 是最小的值。

（1）WHERE C1=5 AND C2=7

两个谓词都是 stage 1 的，复合谓词是 indexable 的。C1 和 C2 作为匹配的列，matching index scan 将被使用。

（2）WHERE C1=5 AND C2>7

两个谓词都是 stage 1 的，复合谓词是 indexable 的。C1 和 C2 作为匹配的列，matching index scan 将被使用。

（3）WHERE C1>5 AND C2=7

两个谓词都是 stage 1 的，但是只有第一个匹配到索引。C1 作为匹配的列，matching index scan 将被使用。

（4）WHERE C1=5 OR C2=7

两个谓词都是 stage 1 的，但不是 Boolean term。复合谓词是 indexable 的。因为没有以 C2 作为前导列的索引，所以多索引访问对复合谓词来说是不可能的。对于单一索引访问，C1 和 C2 是唯一的索引扫描列。

（5）WHERE C1=5 OR C2<>7

第一个谓词是 indexable 的，且是 stage 1 的。第二个谓词是 stage 1 的，但不是 indexable 的。所以复合谓词是 stage 1 的，但不是 indexable 的。

（6）WHERE C1>5 OR C2=7

两个谓词都是 stage 1 的，但不是 Boolean term。复合谓词是 indexable 的。因为没有以 C2 作为前导列的索引，所以多索引访问对复合谓词来说是不可能的。对于单一索引访问，C1 和 C2 是唯一的索引扫描列。

（7）WHERE C1 IN (cor subq) AND C2=C1

两个谓词都是 stage 2 的，且不是 indexable 的。索引不被考虑用于 matching-index access，并且两个谓词都在 stage 2 计算。

（8）WHERE C1=5 AND C2=7 AND (C3+5) IN (7,8)

只用前两个谓词是 stage 1 和 indexable 的。索引被考虑用于 matching-index access，所有满足前两个谓词的记录都被传递到 stage 2 去计算第三个谓词。

（9）WHERE C1=5 OR C2=7 OR (C3+5) IN (7,8)

第三个谓词是 stage 2 的谓词。所以复合谓词也是 stage 2 的，并且所有三个谓词都在 stage 2 计算。简单谓词都不是 Boolean term，并且复合谓词不是 indexable 的。

（10）WHERE C1=5 OR (C2=7 AND C3=C4)

第三个谓词是 stage 2 的。两个复合谓词(C2=7 AND C3=C4)和(C1=5 OR (C2=7 AND C3=C4))也是 stage 2 的。所有的谓词在 stage 2 计算。

（11）WHERE (C1>5 OR C2=7) AND C3=C4

复合谓词(C1>5 OR C2=7)是 indexable 和 stage 1 的。简单谓词 C3=C4 不是 stage 1 的，

所以索引不被考虑用于 matching-index access。满足复合谓词(C1>5 OR C2=7)的记录被传递到 stage 2 用于计算谓词 C3=C4。

2.7.2.11　谓词过滤率

一个谓词的过滤率是介于 0 和 1 之间的数字，用于估计谓词为真的记录占表中全部记录的比例。谓词为真的记录被称为满足那个谓词。

例如：假设 DB2 能够决定表 T 的列 C1 仅仅包含 5 个不同的值：A、D、Q、W 和 X。在不知道其他信息的情况下，DB2 估计五分之一的记录在列 C1 上拥有值 D。那么对于表 T 来说，谓词 C1=‘D’的过滤率为 0.2。

DB2 优化器通过分析 Catalog 中记录数据量及数据分布的数值来估计满足谓词集的记录数量，计算过滤率，进而计算每条访问路径的消耗，选择最优路径。

对简单谓词而言，过滤率是三个变量的函数：

（1）谓词中的值，例如，前一个例子中的‘D’。

（2）谓词中的操作符；例如，前一个例子中的‘=’和这个谓词的反向操作‘<>’。

（3）谓词中列上的统计数据。在前一个例子中，列 T.C1 仅包含 5 个值这样的信息也包含在统计数据中。

建议：当用户编写一个谓词的时候，可以控制前两个变量。

第三个变量的值(列上的统计数据)保存在 DB2 catalog 中。通过运行 RUNSTATS Utility 可以得到更新。

2.7.2.12　简单谓词的默认过滤率

表 2.5 列出了不同类型谓词的默认过滤率。在没有其他统计数据存在的情况，DB2 优化器使用这些值进行路径计算。

例如：谓词 C1=‘D’的默认过滤率为 1/25（0.04）。实际上，如果值 D 的比例不接近 0.04，那么默认值可能就不会产生一个优化的访问路径。

表 2.5　不同谓词类型的默认过滤率

谓词类型	过　滤　率
Col = literal	1/25
Col <> literal	1 - (1/25)
Col IS NULL	1/25
Col IS NOT DISTINCT FROM	1/25

续表

谓词类型	过　滤　率
Col IS DISTINCT FROM	1 - (1/25)
Col IN (literal list)	(number of literals)/25
Col Op literal	1/3
Col LIKE literal	1/10
Col BETWEEN literal1 AND literal2	1/10

注：

Op 是下列操作符之一：<，<=，>，>=。

Literal 是绑定时 DB2 所知道的任何常数值。

2.7.2.13　正态分布的过滤率

DB2 优化器在以下两种情况下使用表 2.6 所示的过滤率：

- Catalog 表 SYSIBM.SYSCOLUMNS 的列 COLCARDF 对于列“Col”有一个正数值。
- Catalog 表 SYSIBM.SYSCOLDIST 中没有对于列“Col”的其他统计信息。

例如：如果 D 是列 C1 的 5 个值中的一个，那么使用 RUNSTATS 后，对于列 C1，表 SYSCOLUMNS 中的列 COLCARDF 的值将为 5。如果没有其他可用的统计信息，那么谓词 C1=‘D’的过滤率为 1/5（0.2）。

表 2.6　不同谓词类型的正态过滤率

谓词类型	过　滤　率
Col = literal	1/COLCARDF
Col <> literal	1 - (1/COLCARDF)
Col IS NULL	1/COLCARDF
Col IS NOT DISTINCT FROM	1/COLCARDF
Col IS DISTINCT FROM	1 - (1/COLCARDF)
Col IN (literal list)	(number of literals)/COLCARDF
Col Op1 literal	interpolation formula
Col Op2 literal	interpolation formula
Col LIKE literal	interpolation formula
Col BETWEEN literal1 AND literal2	interpolation formula

注：

Op1<或者<=，并且 literal 不是宿主变量。

Op2>或者>=，并且 literal 不是宿主变量。

Literal 是绑定时 DB2 所知道的任何常数值。

其他谓词类型的过滤率：前一节的表格和表 2.6 所选择的例子仅是最常用的一些谓词类型。如果 P1 是一个谓词，F 表示它的过滤率，那么谓词 NOT P1 的过滤率为（1-F）。但是过滤率的计算取决于很多东西，所以不能给出所有谓词类型的特定过滤率。

2.7.2.14　DB2 计算过滤率公式

对于使用范围值的谓词来说，DB2 使用 interpolation formula 来计算过滤率。这个公式基于一个估计：谓词中的值数量占表中所有列的值数量的比率。

下面的公式都是粗略的估计，将来可能会被 DB2 更改。它们应用于这种形式的谓词：col op literal。每个公式中 Total ENTRIES 的值，通过 CATALOG 表中 HIGH2KEY 和 LOW2KEY 对于列 Col 的值进行估计：Total Entries = (HIGH2KEY – LOW2KEY)。

- 对于操作符<和<=，此处 literal 不是一个宿主变量：

```
(Literal – LOW2KEY) / (Total Entries)
```

- 对于操作符>和>=，此处 literal 不是一个宿主变量：

```
(HIGH2KEY – Literal) / (Total Entries)
```

- 对于 LIKE 或者 BETWEEN：

```
(High literal – Low literal) / (Total Entries)
```

例如：对于一个谓词的列 C2，假设 HIGH2KEY 的值为 1400，LOW2KEY 的值为 200。对于 C2，DB2 计算得出(Total Entries) = 1200。

对于谓词 C1 BETWEEN 800 AND 1100，DB2 计算得出其过滤率 F 为：

```
F = (1100 – 800)/1200 = 1/4 = 0.25
```

LIKE 谓词过滤率计算公式：DB2 会将 LIKE 谓词作为 BETWEEN 谓词进行处理。谓词中界定范围的两个值产生于谓词中的字符串。只有特定字符（‘%’或者‘_’）前面的前导字符用于生成界定范围的值。所以如果特定字符是字符串中的第一个字符，那么过滤率被估计为 1 且这个谓词被估计不会过滤掉任何记录。

默认过滤率计算原则：在以下情况，DB2 不会进行计算，而是直接使用默认过滤率。

- 仅仅与范围相关，包括 LIKE 和 BETWEEN 谓词。
- 仅当 interpolation 不足的情况下使用。
- 基于 COLCARDF 的值。

- 如果满足下面的条件之一，那么无论正态或者其他的分布数据是否存在于列上都将使用。
 - 谓词不包含常量。
 - COLCARDF < 4。

表 2.7 为操作符<、<=、>、>=、LIKE 和 BETWEEN 列出了过滤率默认值。

表 2.7　默认过滤率

COLCARDF	Op 的过滤率	LIKE 或者 BETWEEN 的过滤率
>=100 000 000	1/10 000	3/100 000
>=10 000 000	1/3 000	1/10 000
>=1 000 000	1/1 000	3/10 000
>=100 000	1/300	1/1 000
>=10 000	1/100	3/1 000
>=1 000	1/30	1/100
>=100	1/10	3/100
>=2	1/3	1/10
=1	1/1	1/1
<=0	1/3	1/10

注：Op 为操作符<，<=，>，>=中的一个。

2.7.2.15　数据分布的过滤率

RUNSTATS 可以为一个列或列的集合生成额外的统计信息。DB2 能够使用这些信息计算过滤率。DB2 收集如下两种分布数据。

- Frequency：对于一个列或者列的集合来说，包含一个值的表中记录的比例。
- Cardinality：列集合中不同值的数量。

表 2.8 列出了使用这些统计数据的谓词类型。

表 2.8　使用统计数据的谓词

Type of statistic	Single column or concatenated columns	Predicates
Frequency	Single	COL=literal COL IS NULL COL IN (literal-list) COL op literal

续表

Type of statistic	Single column or concatenated columns	Predicates
Frequency	Single	COL BETWEEN literal AND literal COL=host variable COL1=COL2 T1.COL=T2.COL COL IS NOT DISTINCT FROM
Frequency	Concatenated	COL=literal COL IS NOT DISTINCT FROM
Cardinality	Single	COL=literal COL IS NULL COL IN (literal-list) COL op literal COL BETWEEN literal AND literal COL=host variable COL1=COL2 T1.COL=T2.COL COL IS NOT DISTINCT FROM
Cardinality	Concatenated	COL=literal COL=:host variable COL1=COL2 COL IS NOT DISTINCT FROM

怎样使用它们：表 SYSCOLDIST 中的列 COLVALUE 和 FREQUENCYF 包含分布的统计数据，可以通过运行 RUNSTATS 实时收集相关信息。

可以以下面的方式运行 RUNSTATS：不使用 FREQVAL 选项，在 correl-spec 中使用 FREQVAL 选项，在 colgroup-spec 中使用 FREQVAL 选项，或者同时使用两种方式。

- 如果不使用 FREQVAL 选项运行 RUNSTATS，那么 RUNSTATS 为特定索引的第一个列中 10 个最经常使用的值插入记录。
- 如果在 correl-spec 中使用 FREQVAL 选项运行 RUNSTATS，那么 RUNSTATS 为一个索引的连起来的多个列插入记录。NUMCOLS 选项说明连起来的索引列的个数。COUNT 选项说明经常使用的值的数量。用户可以收集最经常使用的值、最不经常使用的值或者同时收集两种值。

- 如果在 colgroup-spec 中使用 FREQVAL 选项运行 RUNSTATS，那么 RUNSTATS 为列组中的列插入记录。COUNT 选项说明经常使用的值的数量。用户可以收集最经常使用的值、最不经常使用的值或者同时收集两种值。
- 如果说明了 FREQVAL 选项，RUNSTATS 为特定索引的列和列组中的列插入记录。

下面举两个例子说明如何计算过滤率。

例子 1：单个列的过滤率

假设谓词为 C1 IN (‘3’，‘5’)，且 SYSCOLDIST 为列 C1 包含如下值：

```
COLVALUE    FREQUENCYF
  '3'         0.0153
  '5'         0.0859
  '8'         0.0627
```

过滤率是 0.0153 + 0.0859 = 0.1012。

例子 2：相关列的过滤率

假设列 C1 和 C2 是相关的。假设谓词为 C1=‘3’ AND C2=‘5’，并且 SYSCOLDIST 为列 C1 和 C2 包含如下的值：

```
COLVALUE       FREQUENCYF
 '1'  '1'         0.1176
 '2'  '2'         0.0588
 '3'  '3'         0.0588
 '3'  '5'         0.1176
 '4'  '4'         0.0588
 '5'  '3'         0.1764
 '5'  '5'         0.3529
 '6'  '6'         0.0588
```

过滤率是 0.1176。

2.7.2.16 使用多个过滤率决定一个查询的消耗

当 DB2 估计一个查询消耗时，它在各个层级逐层决定过滤率。例如，假设执行了下面的查询：

```
SELECT COLS FROM T1
  WHERE  C1 = 'A'
    AND  C3 = 'B'
```

```
    AND  C4 = 'C';
```

表 T1 由列 C1、C2、C3 和 C4 组成。索引 I1 定义在表 T1 上，并且由列 C1、C2 和 C3 组成。

假设复合谓词中的简单谓词有如下属性：

- **C1 = ‘A’**　　Matching predicate
- **C3 = ‘B’**　　Screening predicate
- **C4 = ‘C’**　　Stage 1，nonindexable predicate

为了决定通过索引 I1 访问表 T1 的消耗，DB2 实施下面的步骤：

（1）估计 matching index 的消耗。因为 matching column 只能是一个 column，所以 DB2 使用单个列的 cardinality 和单个列的 frequency 数据决定 matching index 的过滤率。

（2）估计所有索引的过滤率。这包括 matching 和 screening 索引的过滤率。如果在列组(C1,C3)上存在统计信息，DB2 将使用这些信息。否则 DB2 使用这些列上的、可利用的单列统计信息。

DB2 也会使用 FULLKEYCARDF 作为一个边界限制条件。因此，为了得到精确的估计，在列组(C1,C3)上拥有列组的统计信息是非常关键的。

（3）估计表级别的过滤率。如果在列组(C1,C3)上存在统计信息，DB2 将使用这些信息。否则，DB2 使用存在于这些列的子集上的统计信息。

如果在过滤的每个层级都提供了恰当的统计数据，那么 DB2 选择最有效率访问路径的可能性会大大增加。用户可以使用 RUNSTATS 收集所需要的任何统计数据。

2.7.2.17　列相关

数据的两个列（比如一个表的列 A 和 B）是相关的，如果列 A 的值不能独立于列 B 的值进行变化。

例子：表 2.9 列出了一张大表的部分数据。列 CITY 和 STATE 是高度相关的，但是列 DEPNO 和 SEX 是完全独立的。

表 2.9　列相关示例

CITY	STATE	DEPTNO	SEX	EMPNO	ZIPCODE
Fresno	CA	A345	F	27375	93650
Fresno	CA	J123	M	12345	93710
Fresno	CA	J123	F	93875	93650
Fresno	CA	J123	F	52325	93792

续表

CITY	STATE	DEPTNO	SEX	EMPNO	ZIPCODE
New York	NY	J123	M	19823	09001
New York	NY	A345	M	15522	09530
Miami	FL	B499	M	83825	33116
Miami	FL	A345	F	35785	34099
Los Angeles	CA	X987	M	12131	90077
Los Angeles	CA	A345	M	38251	90091

在这个简单的例子中，对于值为‘FRESNO’的列 CITY 的每个值，在 STATE 列都存在一个相同的值‘CA’。

2.7.3 DB2 谓词管理

在某些特定情况下，DB2 优化器会在 SQL 语言解析时会对用户编码的谓词进行修改，或者产生额外的谓词。这些改变对用户是透明的，但会影响访问路径的选择。这是因为在消耗较低的情况下，DB2 总是使用索引访问路径。生成的额外谓词提供了更多的、潜在的 indexable predicate，这也为使用高效的索引访问路径创造了更多的机会。

下面将介绍 DB2 是怎样管理谓词的。

2.7.3.1 IN-list谓词的修改

如果一个 IN-list 谓词在它的列表中仅有一项，那么 DB2 会将这个谓词变成 EQUAL 谓词。通过 OR 连接起来的、相同列上的多个简单的、Boolean term 的相等谓词可以被转化成 IN-list 谓词。例如：C1=5 or C1=10 or C1=15 将转化为 C1 IN (5,10,15)。

2.7.3.2 简化JOIN操作

因为全外部连接（full outer join）不如左连接或右连接高效，并且左连接和右连接不如内部连接高效，所以在查询中应该总是尽量使用类型最简单的连接操作。然而，如果遇到了可以简化的连接，DB2 将试图简化。一般来说，当查询中包含消除连接操作产生的 NULL 值的谓词或者 ON 从句时，DB2 可以简化连接操作。下面使用一个示例进行说明：

```
SELECT * FROM T1 X FULL JOIN T2 Y
  ON X.C1=Y.C1
  WHERE X.C2 > 12;
```

正常执行外部连接操作会有这样的结果表记录：

- 表 T1 和 T2 中列 C1 值匹配的记录（内部连接结果）。
- 列 C1 在 T2 中没有相应值的 T1 表中的记录。
- 列 C1 在 T1 中没有相应值的 T2 表中的记录。

然而，在 DB2 解析时会判断关于“列 C1 在 T1 中没有相应值的 T2 表中的记录”。这些记录对于这句 SQL 是无意义的，因此 DB2 将全连接转化为更加高效的左连接，直接将 SQL 语言改写成以下形式：

```
SELECT * FROM T1 X LEFT JOIN T2 Y
  ON X.C1=Y.C1
  WHERE X.C2 > 12;
```

在 DB2 将外连接转化为一个简单的外连接或者一个内连接之前，紧跟连接操作后面的谓词一定具有下面的属性：

- 这个谓词是一个 Boolean term 谓词。
- 如果连接操作中的一张表为所有列产生了 NULL 值，那么这个谓词为 false。

下面的谓词示例都是能够使 DB2 简化连接操作的例子：

- T1.C1 > 10
- T1.C1 IS NOT NULL
- T1.C1 > 10 OR T1.C2 > 15
- T1.C1 > T2.C1
- T1.C1 IN (1,2,4)
- T1.C1 LIKE ‘ABC%’
- T1.C1 BETWEEN 10 AND 100
- 12 BETWEEN T1.C1 AND 100

2.7.3.3　通过隐含关系产生新谓词

当一个查询中的谓词集合在逻辑上隐含其他谓词的时候，DB2 能够生成额外的谓词，为访问路径的选择提高更多的信息。

生成谓词的原则如下。

（1）对于单表或者内连接查询而言，有下列几种情况 DB2 会根据隐含关系生成新的谓词。

查询在第一个谓词中的一列上有一个 Boolean term 谓词，且谓词具有下面的形式：

- COL1 op value

 op 是=，<>，>，>=，<或者<=。

 value 是一个常量、宿主变量或者特殊寄存器。

- COL1 (NOT) BETWEEN value1 AND value2
- COL1 = COL3

（2）对于外部连接查询来说，如果这个查询有一个 COL1=COL2 形式的 ON 从句，并且有一个如下形式的连接前谓词时 DB2 会生成新的谓词。

- COL1 op value。

 op 是=，<>，>，>=，<或者<=。

- COL1 (NOT) BETWEEN value1 AND value2。

当且仅当生成的谓词不涉及带有不匹配记录的表时，DB2 才为外部连接查询生成一个隐含关系谓词。也就是说，生成的谓词不能涉及左外部连接的左表或右外部连接的右表。

（3）对于一个多 CCSID 查询，只有当生成的谓词具有下列属性时，DB2 才会根据隐含关系生成新的谓词。

- 生成的谓词是一个范围谓词（op 是>，>=，<或者<=）。
- 带有生成谓词的查询的计算导致与不带有这个谓词的查询的计算之间不同的 CCSID 的转换。

当一个谓词满足了隐含关系的条件时，无论在 WHERE 从句中是否已经存在，DB2 都会生成一个新的谓词。

生成的谓词具有下面的形式：

- COL op value

 op 是=，<>，>，>=，<或者<=。

 value 是一个常量、宿主变量或特殊寄存器。

- COL (NOT) BETWEEN VALUE1 AND VALUE2。
- COL1=COL2（仅对单表或者内连接有效）。

下面举例介绍隐含关系的谓词改写。假设用户已经编写了下面这个查询，这个查询满足根据隐含关系改写 SQL 的条件：

```
SELECT * FROM T1, T2
  WHERE T1.C1=T2.C1 AND
T1.C1>10;
```

DB2 会生成一个额外的谓词以产生一个更加高效的查询：

```
SELECT * FROM T1, T2
  WHERE T1.C1=T2.C1 AND
  T1.C1>10 AND
  T2.C1>10;
```

谓词冗余：当查询中其他谓词的计算已经决定了这个谓词提供的结果时则这个谓词就是冗余的。用户可以说明冗余谓词或者由 DB2 生成。DB2 不决定查询谓词中的任何一个是冗余的。无论是否冗余，用户编写的所有谓词都在执行时进行计算。如果为了有助于选择访问路径，DB2 生成了冗余谓词，那么谓词将在执行时被忽略。

增加额外的谓词：DB2 仅在相等（equal）和范围（range）谓词上进行谓词改写。然而，通过为其他的操作类型（例如 IN 或者 LIKE）增加额外的谓词，可以帮助 DB2 选择一个更好的访问路径。例如，考虑下面的 SELECT 语句：

```
SELECT * FROM T1,T2
  WHERE T1.C1=T2.C1
  AND T1.C1 LIKE 'A%';
```

如果 T1.C1=T2.C1 为真，并且 T1.C1 LIKE ‘A%’为真，那么 T2.C1 LIKE ‘A%’必定也为真。因此，通过增加 T2.C1 LIKE ‘A%’，可以为计算这个查询为 DB2 提供额外的信息：

```
SELECT * FROM T1,T2
  WHERE T1.C1=T2.C1
  AND T1.C1 LIKE 'A%'
  AND T2.C1 LIKE 'A%';
```

2.7.3.4　数据加密情况下的谓词

DB2 为数据加密和解密提供了内建的函数。这些函数可以保护敏感的数据，但是在使用中也会降低某些语句的性能。如果一个谓词包含=和<>以外的其他操作符，那么实施比较前，加密的数据一定要被解密。解密将使谓词变为 stage 2 的。

2.7.4　高效地使用宿主变量

宿主变量需要默认的过滤率：当用户绑定一个包含宿主变量的静态 SQL 语句时，DB2 使用默认的过滤率来为这个 SQL 语句决定最好的访问路径。

对于一个带有几个宿主变量的查询，通常情况下，DB2 都会选择表现较好的访问路径。

然而，在发布新的 DB2 版本后或者实施维护后，DB2 可能选择一个新的访问路径，而这个访问路径不如旧的访问路径表现好。在许多情况下，访问路径的改变都是由于默认过滤率引起的。默认过滤率可能以一种不同的方式引导 DB2 优化查询。

对于一个包含宿主变量的查询来说，改变访问路径的方法有以下两种：

（1）绑定包含这个查询的包或者计划时，使用 REOPT（ALWAYS)或者 REOPT(ONCE)选项。

（2）重新编写这个查询。

2.7.4.1 运行时改变访问路径

为了控制 DB2 如何决定带有变量值 SQL 语句的访问路径，可以使用下面的绑定选项：

- REOPT(ALWAYS)

每次 SQL 语句运行时，DB2 都为带有变量值的任何 SQL 语句决定访问路径。

- REOPT(ONCE)

SQL 运行时，DB2 使用第一次输入的变量值，为带有变量值的任何 SQL 语句决定和缓存访问路径仅仅一次。如果这个语句运行多次，DB2 不会每次都优化，除非缓存的语句已经无效或者已经从缓存中删除。

- REOPT(NONE)

DB2 在绑定时决定访问路径，并且运行时不改变访问路径。

2.7.4.2 重写查询以影响访问路径的选择

下面通过一个实例来说明如何通过改写 SQL 语句来影响访问路径。然而，在考虑重新编写任何一个查询之前，都应该先考虑使用 REOPT(ALWAYS)或者 REOPT(ONCE)选项是否能够解决访问路径问题。

例子：一个相等谓词（equal predicate）

一个相等谓词有一个值为 1/COLCARDF 的默认过滤率。但是，实际的过滤率可能是非常不同的。

查询语句如下：

```
SELECT * FROM DSN8810.EMP
     WHERE SEX = :HV1;
```

假设：因为列 SEX 仅有两个不同的值，即‘M’和‘F’，所以 SEX 的 COLCARDF

的值为 2。如果男性和女性雇员的数量不是相等的，取决于:HV1 被设置为‘M’还是‘F’，实际的过滤率会大于或者小于默认的过滤率。

建议：下面的两个措施会改善访问路径。

使用 REOPT(ALWAYS)选项，绑定包含这个查询的包或者计划。这将使 DB2 使用用户提供的输入值，在运行时重新优化查询。用户也可以考虑使用 REOPT(ONCE)选项来绑定包或者计划。

基于对实际过滤率的了解，编写可以影响 DB2 选择访问路径的谓词。例如，可以将这个查询分解成三个不同的查询，其中两个使用常量。然后当绑定计划时，DB2 就能够决定准确的过滤率。

```
SELECT (HV1);
   WHEN ('M')
      DO;
        EXEC SQL SELECT * FROM DSN8810.EMP
          WHERE SEX = 'M';
      END;
   WHEN ('F')
      DO;
         EXEC SQL SELECT * FROM DSN8810.EMP
          WHERE SEX = 'F';
      END;
   OTHERWISE
      DO:
        EXEC SQL SELECT * FROM DSN8810.EMP
         WHERE SEX = :HV1;
      END;
END;
```

2.7.5　编写高效的子查询

子查询是一个 SELECT 语句，这个语句位于另外一个 SQL 语句的 WHERE 或者 HAVING 从句中。

通常情况下，子查询语句可以通过编写两个或者更多的 SQL 语句得到同样的结果。然而，这些语句有不同的访问路径，并且效率可能也不一样。具体采用哪种语言实现需要用户根据情况进行考虑。

2.7.5.1 相关子查询

一个相关子查询是指涉及外部查询的至少一个列。

包含相关子查询的任何一个谓词是一个 stage 2 的谓词，除非它被转化成 JOIN。

下面的查询展示了涉及外部查询的子查询：

```
SELECT * FROM DSN8810.EMP X
   WHERE  JOB = 'DESIGNER'
     AND  EXISTS (SELECT 1
                  FROM   DSN8810.PROJ
                  WHERE  DEPTNO = X.WORKDEPT
                     AND  MAJPROJ = 'MA2100');
```

当外部查询中每条满足条件的记录被读取的时候，相关子查询被计算。在执行这个查询的过程中，DB2 的处理步骤是这样的：

STEP 01 从表 EMP 中读取一条满足 JOB=‘DESIGNER’的记录。

STEP 02 在一个存储在内存中的表中，搜寻那条记录中 WORKDEPT 的值。

内存中的表保存子查询的执行结果。若果这个子查询针对 WORKDEPT 的这个值已经执行过，那么子查询的结果就在表中，并且针对这条记录 DB2 不会再次执行子查询。取而代之，DB2 将跳到步骤 5。

STEP 03 如果 WORKDEPT 的值不在内存中，则执行子查询。这要求搜寻表 PROJ，检查是否存在 MAJPROJ 为 MA2100 的任何 project，而且对于这个 project 来说，当前的 WORKDEPT 是合理的。

STEP 04 将 WORKDEPT 的值和子查询的结果存储在内存中。

STEP 05 将 EMP 中当前记录的值返回给应用。

对于 EMP 表中每条满足条件的记录，DB2 重复执行整个过程。

2.7.5.2 不相关子查询

一个不相关子查询与外部查询是一样的。

下面的 SQL 语句是一个不相关查询的例子。

```
SELECT * FROM DSN8810.EMP
       WHERE  JOB = 'DESIGNER'
         AND  WORKDEPT IN (SELECT DEPTNO
                           FROM   DSN8810.PROJ
                           WHERE  MAJPROJ = 'MA2100');
```

当游标为这个查询被打开时，不相关子查询被执行一次。DB2 怎样处理它取决于它返回一条记录还是多条记录。前一个例子中的查询可以返回多条记录。

2.7.5.3　单值子查询

当子查询被包含在使用简单操作符的谓词中时，这个子查询被要求返回 1 条或者 0 条记录。简单操作符可以是下面操作符中的一个：

<, <=, >, >=, =, <>, NOT <, NOT <=, NOT >, NOT >=

下面的不相关子查询返回一个简单值：

```
SELECT *
  FROM   DSN8810.EMP
  WHERE  JOB = 'DESIGNER'
    AND  WORKDEPT <= (SELECT MAX(DEPTNO)
                  FROM   DSN8810.PROJ);
```

当游标被打开的时候，执行字查询。如果它返回多条记录，DB2 将会报错。包含这个子查询的谓词被像使用了常量的简单谓词（例如，WORKDEPT <= ‘value’）一样处理。

通常情况下，决定带有返回单值不相关子查询的谓词是 stage 1 还是 stage 2 的规则，与带有单个变量的同样查询是一样的。

2.7.5.4　多值子查询

一个子查询可以返回多条记录，如果操作符为下面操作符的其中之一：

```
op ANY, op ALL , op SOME, IN, EXISTS
```

此处，op 为这些操作符：>, >=, <, <=, NOT <, NOT <=, NOT >, NOT >=。

可能的情况下，DB2 将返回多条记录的子查询调整为仅返回一条记录。当有一个带有 ANY、ALL 或者 SOME 的范围比较时，将发生这种情况。下面的查询是一个例子：

```
SELECT * FROM DSN8810.EMP
   WHERE  JOB = 'DESIGNER'
     AND  WORKDEPT <= ANY (SELECT DEPTNO
                      FROM   DSN8810.PROJ
                      WHERE  MAJPROJ = 'MA2100');
```

DB2 计算表 DSN8810.PROJ 中 DEPTNO 的最大值，并且移除查询中的 ANY 关键字。在这个转换后，这个子查询被像一个单值子查询一样处理。

可以使用最大值进行这个转换，如果范围操作符为：

- 带有 ALL 关键字的>或者>=。
- 带有 ANY 或者 SOME 关键字的<或者<=。

可以使用最小值进行这个转换，如果范围操作符为：

- 带有 ALL 关键字的<或者<=。
- 带有 ANY 或者 SOME 关键字的>或者>=。

与带有单值子查询的同样谓词的规则一样，结果谓词被决定为 stage 1 或者 stage 2 的。

2.7.5.5 将子查询转换成JOIN操作的条件

对于 SELECT、UPDATE 或者 DELETE 语句来说，DB2 有时能够将一个子查询转化为一个 JOIN，这个 JOIN 是子查询的结果表和外部查询的结果表之间的 JOIN。

对于一个 SELECT 语句来说，如果满足下面的条件，DB2 实施转化：

- 转化没有增加冗余度。
- 子查询出现在 WHERE 从句中。
- 子查询不包括 GROUP BY、HAVING 或者聚集函数。
- 子查询的 FROM 从句中仅有一张表。
- 对于一个相关子查询来说，含有子查询的谓词中的比较操作符为 IN、EXISTS、=ANY、或者=SOME。
- 对于一个非相关子查询来说，谓词的左边为与子查询的列具有相同数据类型和长度的单个列（对于相关子查询，左边可以是任何一个表达式）。

对于 UPDATE、DELETE 语句或者不满足上面条件的 SELECT 语句来说，如果满足下面的条件，那么 DB2 将一个相关子查询转化为 JOIN：

- 转化没有增加冗余度。
- 子查询被相关到它紧邻的外部查询。
- 子查询的 FROM 字句中仅包含一张表，并且外部查询（针对 SELECT）、UPDATE 或 DELETE 仅涉及一张表。
- 如果一个外部谓词是一个使用=ANY 操作符的谓词或 IN 谓词，下面的条件为真：
 - — 外部谓词的左边是一个单列。
 - — 外部谓词的右边是一个涉及单列的子查询。
 - — 两列有同样的数据类型和长度。
- 子查询不包含 GROUP BY 和 DISTINCT 从句。

- 子查询不包含聚集函数。
- 子查询的 SELECT 从句不包含一个用户定义的函数。
- 子查询谓词是一个 Boolean term 谓词。
- 提供相关性的子查询中的谓词是 stage 1 的谓词。
- 子查询不包含嵌套式子查询。
- 子查询不包含访问自己的 UPDATE 或者 DELETE。
- 对于一个 SELECT 语句，查询中不包含 FOR UPDATE OF 从句。
- 对于一个 UPDATE 或者 DELETE 语句，语句是一个搜索式的 UPDATE 或者 DELETE。
- 对于 SELECT 语句，没有启用并行。

对于带有多个子查询的语句来说，DB2 仅在满足转化条件的最后一个子查询上实施转化。

2.7.5.6　子查询调优

下面的三个查询都获取同样的记录。这三个查询都获取部门中关于所有设计师的数据，这些设计师负责主要工程 MA2100 的一部分。这三个查询展示了：为了获取一个期望结果，可以有多个不同的方法。

- 查询 A：两个表的 JOIN

```
SELECT DSN8810.EMP.* FROM DSN8810.EMP, DSN8810.PROJ
    WHERE  JOB = 'DESIGNER'
      AND  WORKDEPT = DEPTNO
   AND  MAJPROJ = 'MA2100';
```

- 查询 B：一个相关子查询

```
SELECT * FROM DSN8810.EMP X
    WHERE  JOB = 'DESIGNER'
      AND  EXISTS (SELECT 1 FROM DSN8810.PROJ
                   WHERE  DEPTNO = X.WORKDEPT
                  AND  MAJPROJ = 'MA2100');
```

- 查询 C：一个非相关子查询

```
SELECT * FROM DSN8810.EMP
     WHERE  JOB = 'DESIGNER'
       AND  WORKDEPT IN (SELECT DEPTNO FROM DSN8810.PROJ
                  WHERE  MAJPROJ = 'MA2100');
```

在输出中，如果需要来自表 EMP 和 PROJ 中的列，则必须使用一个 JOIN。

子查询中，PROJ 可能包含 DEPTNO 的重复值，所以不能编写一个等价的 JOIN。

一般来说，查询 A 可能是效率最高的。然而，如果表 PROJ 中的列 DEPTNO 上没有索引，那么查询 C 可能是效率最高的。查询 C 中的 IN 子查询谓词是 indexable 的。因此，如果存在一个列 WORKDEPT 上的索引，在表 EMP 上，DB2 可能实施 IN-list 访问。如果决定不使用 JOIN，并且表 PROJ 中的列 DEPTNO 上有一个可用的索引，那么查询 B 可能是效率最高的。

当查看一个有问题的子查询时，看看这个查询是否能够被重写成另外一种格式，或者看看是否可以新建一个索引来帮助改善子查询的效率。

对于不同的子查询谓词和查询中所有其他的谓词来说，知道计算的顺序是非常重要的。如果一个子查询谓词消耗很大，或许其他的谓词能够在这个谓词之前进行计算，所以在计算有问题的子查询谓词之前就将剔除那些记录。

2.7.6 如何判断一个SQL有问题

前面几节讲述了 SQL 调优的基本方法，但是如何判断一个 SQL 的效率有问题呢？

1．打印DB2 性能监控报表

跟踪性能的其中一种方法是打印 DB2 性能监控报表，包括 accounting report、statistics report、SQL activity report 等。当包含 SQL 语句的程序以提交作业的方式执行，这些作业已经运行完成，而用户怀疑程序有问题时，此种方法尤其适用。

2．使用绑定选项EXPLAIN

当用户绑定一个计划或者包时，可以使用 EXPLAIN 选项。使用 EXPLAIN 选项绑定计划或者包后，通过查看 PLAN_TABLE，就可以看到 SQL 的访问路径信息。如果新的访问路径为 table space scan 或者 nonmatching index space scan，但是旧的访问路径没有使用这些方式，就说明这个 SQL 语句可能有问题。

3．使用PM工具

PM 工具是 DB2 提供的一个联机性能监控工具。使用此工具可以监控 DB2 系统中正在运行的 SQL 语句是否有问题。

2.7.7　小结

本书所提供的调优建议都是一些通用性的建议。在实际的调优过程中，调优的措施和手段高度依赖于工作负载和 DB2 之外的系统属性。

DB2 系统仅仅是整个系统的一部分。ZSeries 硬件、磁盘子系统、z/OS、IMS、CICS、TCP/IP、VTAM 等的任何改变，都将影响 DB2 和建立在其上的应用程序的运行效率。

2.8　DB2 安全控制与审计

数据库管理员的一个很重要的职责是保证数据库的安全性。本节详细讲解 DB2 内部安全控制机制原理，以及如何通过 DB2 安全控制，实现对数据库对象的保护，防止对数据的未经授权的访问，未经授权的人对数据进行插入、删除、修改及如何监控用户对数据的存取。有了安全控制，就需要有机制确保是否严格按照安全策略执行，因此就需要审计。本节后面会对 DB2 安全审计做简单介绍。

2.8.1　数据库安全控制范围

数据库管理员的一个很重要的职责是保证数据库的安全性，而数据库安全重点关注的任务主要有如下几方面：

（1）防止软硬件故障造成的意外的数据丢失或损害数据库的完整性

对于此类事故造成的数据恢复可由数据库的日志配合备份策略来完成。

（2）防止对数据的未经授权的访问

要防止信息被不需要知道的人获取。

（3）防止未经授权的人对数据进行插入、删除、修改

要防止未经授权的人通过删除或者篡改数据进行破坏。

（4）监控用户对数据的存取

总的来说，数据库管理员要制订完善的安全性计划，包括定义数据库存取控制计划的目标，指定哪些人在什么情况下可存取哪些内容。

2.8.2 DB2 安全控制简介

DB2 的安全控制，主要是指对 DB2 子系统、DB2 内存储的数据或对象的存取控制。系统管理员通过指定一个安全计划设置安全访问的目标，决定谁能在何种情况下访问何种资源。安全计划还描述通过何种手段（DB2 的程序、其他系统程序、或管理过程等）达到安全访问的目标。

DB2 在安全方面采用两层验证机制，第一层为身份验证（Authentication），就是利用操作系统来验证用户的密码是否正确；第二层被称为存取控制授权（Access Control Authorization），主要由数据库管理器来验证用户有没有执行具体操作的权限。

DB2 定义了一个权限层次结构，用于将一组预先确定的管理权限授予用户或用户组。这些管理权限包括能够对数据库进行备份，更改数据库配置参数，查看数据表中的内容等。

DB2 对以上权限的控制主要是通过 ID（Identifiers）来实现的，而 ID 主要有如下四类：

（1）Primary Authorization ID

Primary Authorization ID 是作为资源访问控制的主要标志，例如 Statistics 和 Performance Trace 就是使用 Primary Authorization ID 来进行标识的。

（2）Secondary Authorization ID

Secondary Authorization ID 是一类对资源访问有辅助权限控制的 ID，例如 RACF 用户组 ID，就是 Secondary authorization ID。

（3）SQL ID

在 DB2 中，默认的 SQLID 是用户登录系统的 ID。而用户也可以使用 SET CURRENT SQLID=‘XYZ’来更改当前的 SQLID。当前用户如果拥有 SYSADM 权限，可以设置任何值为当前 SQLID。

- 在完成 SQLID 的设置后进行的任何 SQL 语句操作，都使用设的值作为权限验证 ID。
- 使用 CREATE 语句进行创建表空间，创建 DATABASE，创建 STORAGE GROUP 等操作后，对应的 OWNER 即为使用 SET 语句设置的 SQLID。
- 使用 CREATE 语句进行数据表，视图（VIEW），别名（ALIAS）等创建后，对应的 QUALIFER 也是使用 SET 语句设置的 SQLID（是在动态 SQL 运行的时候使用的 ID，

可以使用 Set Current SQLID 的方式将 SQLID 置为 Primary authorization ID 或者 Secondary Authorization ID。如果执行 Set 语句的用户拥有 SYSADM 权限，那么可以将 SQLID 置为任意一个 ID）。

（4）RACF ID

通常来说，RACF ID 就是 Primary Authorization ID，如果你的系统启用了 RACF，那么你登录系统使用的 ID 就是 RACF ID，如图 2.22 所示。

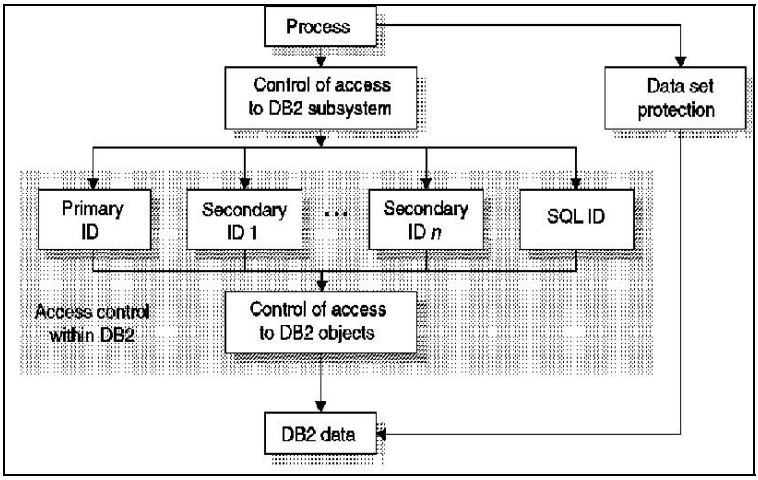


图 2.22　DB2 安全机制的 ID 控制

当用户通过了用户标识和鉴定，成功登录系统后，对数据库中的数据进行访问的时候，就要根据预先定义好的用户权限进行存取控制，保证用户只能存取他有权存取的数据。这里的用户是指不同的用户对于不同的数据对象允许执行的操作权限，它由两部分组成，一是数据对象，二是操作类型。定义用户的存取权限就是要设置该用户可以在哪些数据对象上进行哪些类型的操作。

在对用户数据存取进行授权的时候，数据库管理员要综合考虑授权粒度，粒度指可以定义的数据对象的范围。

- 它是衡量授权机制是否灵活的一个重要指标。
- 授权定义中数据对象的粒度越细，即可以定义数据对象的范围越小，授权子系统就越灵活，系统定义与检查权限的开销会相应增大。
- 能否提供与数据值有关的授权反映了授权子系统精巧程度。

2.8.3 DB2 对象访问控制

DB2 通过一系列的权限设定来对各类对象进行访问控制，每一类权限控制了某些特定对象的访问权限，该权限是指对特定的对象允许执行特定的功能。DB2 中的权限分为显性权限和隐性权限。

- 显性权限：通过 SQL GRANT 授予的权限。
- 隐性权限：对象的所有者对它所创建的对象拥有一系列权限。

作为数据库管理员，在对所管理的数据进行安全保护策略制定之前，必须了解数据访问控制的几种典型方法，下文将通过详细介绍显性权限和隐性权限来帮助数据库管理员进行安全策略的规划。

2.8.3.1 显性权限

为了控制对各类资源的访问，数据库管理员可以通过授权，不授权及取消授权几种方式实现。

显性权限是一类特定的权限，每种显性权限有对应的名称，是可以通过 GRANT 语句进行授权，可以通过 REVOKE 语句进行权限的取消。例如选择（SELECT）权限。

2.8.3.2 如何获取显性权限

用户可以使用 SQL DCL 的 GRANT 语句对 DB2 的各类对象进行授权，完成授权后此用户对此对象所拥有的权限有完全的控制权，也可以进行回收授权等操作。下面将列举几种常用的显性权限，如表 2.10～表 2.14 所示。

表 2.10 Collection 的显性权限

Collection 权限	对于某一确定的 collection 可允许的操作
CREATE IN	BIND PACKAGE 操作

表 2.11 DATABASE 的显性权限

Database 权限	对于某一确定的 DATABASE 可允许的操作
CREATETAB	对某一 Database 执行 CREATE TABLE 语句，创建表
CREATETS	对某一 Database 执行 CREATE TABLESPACE 语句，创建表空间
DISPLAYDB	执行 DISPLAY DATABASE 命令，显示 Database 状态
DROP	对某一 Database 执行 DROP 或 ALTER DATABASE 操作，删除或修改 Database 属性

续表

Database 权限	对于某一确定的 DATABASE 可允许的操作
IMAGCOPY	执行 QUIESCE, COPY, MERGECOPY MODIFYutilities
LOAD	执行 LOAD Utility 向表中装载数据
RECOVERDB	执行 RECOVER, REBUILD INDEX, REPORT utilities
REORG	执行 REORG Utility
REPAIR	执行 REPAIR 和 DIAGNOSE utilities（除 REPAIR DBD 和 DIAGNOSE WAIT）
STARTDB	执行 START DATABASE 命令
STATS	执行 RUNSTATS, CHECK, LOAD, REBUILD INDEX,REORG INDEX, 和 REORG TABLESPACE utilities
STOPDB	执行 STOP DATABASE 命令

表 2.12　PACKAGE 的显性权限

Package 权限	对某一确定的 package 可允许的操作
BIND	执行 BIND, REBIND, FREE PACKAGE 命令
COPY	在 BIND PACKAGE 时指定 COPY 选项以复制一个 package
EXECUTE	执行在一个 PLAN 的 PKGLIST 中包含的 Package
GRANT ALL	所有 Package 相关的权限

表 2.13　PLAN 的显性权限

Plan 权限	对某一确定的应用 Plan 可允许的操作
BIND	执行 BIND, REBIND, FREE PLAN 命令
EXECUTE	在运行一个应用程序时执行对应的 Plan

表 2.14　DB2 系统的显性权限

系统 privilege	DB2 系统操作权限
ARCHIVE	执行 ARCHIVE LOG 命令，归档当前活动日志； 执行 DISPLAY ARCHIVE 命令，显示归档日志信息； 执行 SET LOG 命令，修改系统 checkpoint 频度； 执行 SET ARCHIVE 命令，控制日志归档过程中的磁带设备的分配和释放；
BINDADD	允许用户在 BIND 的命令中使用 ADD 选项，创建新的 Plan 和 Package
BINDAGENT	执行 BIND, REBIND, FREE 命令，DROP PACKAGE 语句，以授权者的名义 bind, rebind，或 free plan、package，或复制一个 package。

续表

系统 privilege	DB2 系统操作权限
BSDS	执行 RECOVER BSDS 命令，恢复 bootstrap 文件集
CREATEALIAS	执行 CREATE ALIAS 语句创建表或视图的别名
CREATEDBA	执行 CREATE DATABASE 语句创建一个 database 并且对其具有 DBADM 权限
CREATEDBC	执行 CREATE DATABASE 语句创建一个 database 并且对其具有 DBCTRL 权限
CREATESG	执行 CREATE STOGROUP 语句创建一个 storage group
CREATETMTAB	执行 CREATE GLOBAL TEMPORARY TABLE 语句，创建一个临时表
DISPLAY	执行 DISPLAY ARCHIVE, DISPLAY BUFFERPOOL, DISPLAY DATABASE, DISPLAY LOCATION, DISPLAY LOG, DISPLAY THREAD, DISPLAY TRACE 命令显示系统信息
MONITOR1	收集不是特殊敏感类型的 trace 数据
MONITOR2	收集所有的 trace 数据
RECOVER	执行 RECOVER INDOUBT 命令恢复线程
STOPALL	执行 STOP DB2 命令，停止 DB2 子系统
STOSPACE	执行 STOSPACE Utility，收集空间使用情况信息
TRACE	执行 START TRACE, STOP TRACE, MODIFY TRACE 控制 DB2 TRACE

2.8.3.3 管理权限

管理权限（Administrative authority）是指一系列权限的集合，可以执行某一类特定的操作，此类操作不是显性的，可能没有名称，或者是并不能通过授权语句进行授权的操作。例如一个 ID 被授予了 SYSOPR 管理权限后，这个 ID 即隐性地被授予了可以终止 Utility 的权限。图 2.23 显示了各种管理权限用户所拥有的权限及各自之间的关系。

下面详细介绍几类常用的管理权限。

【SYSOPR】

SYSOPR 指数据库系统操作员，拥有如下权限：

- 有权限执行大部分的 DB2 命令。
- 不能执行诸如 ARCHIVE LOG、START DATABASE、STOP DATABASE 和 RECOVER BSDS 等命令。
- 可以终止运行的 DB2 Utility。

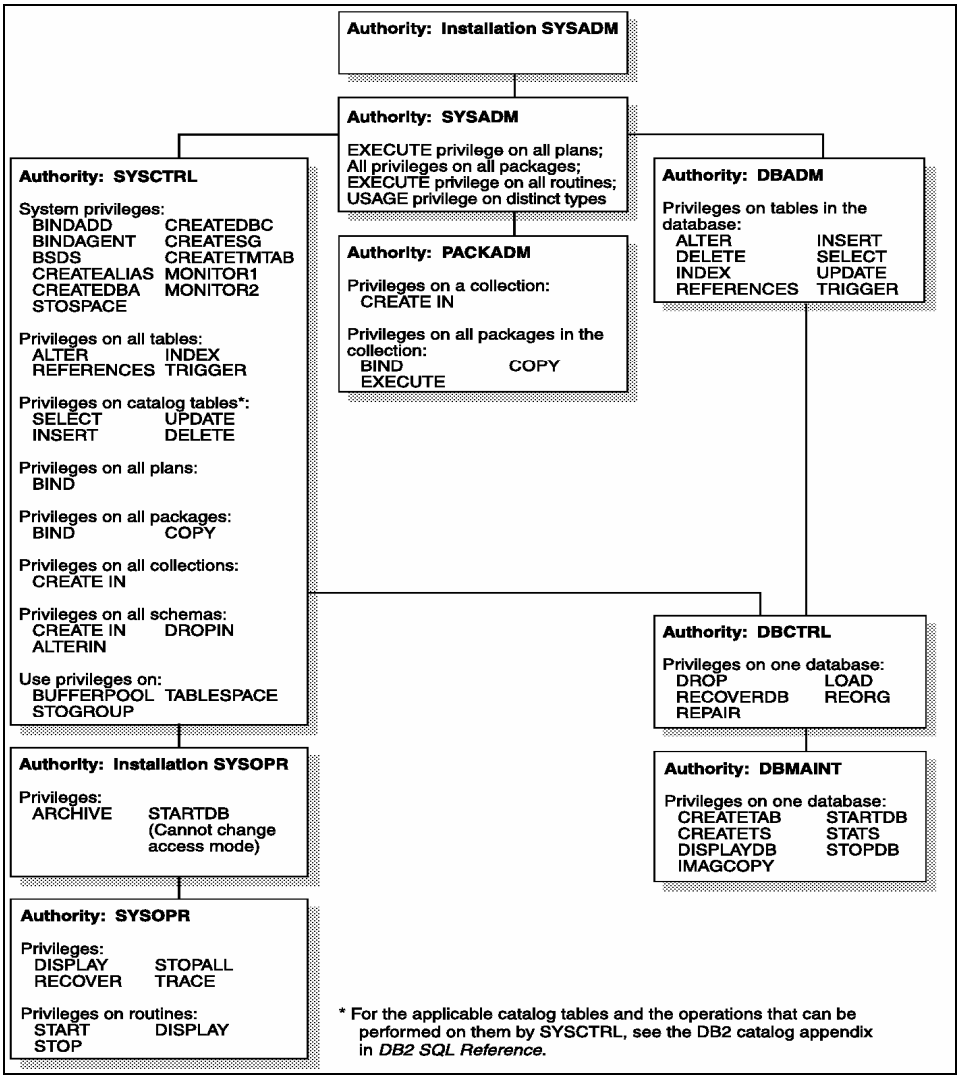


图 2.23　管理权限概览

- 可以运行 DSN1SDMP Utility。
- 可以对其他用户授予 SYSOPR 的权限。

【Installation SYSOPR】

在进行 DB2 子系统安装的时候，可以指定 1～2 个用户 ID 作为 Installation SYSOPR，其在拥有 SYSOPR 所有权限的同时，还拥有如下权限：

- Installation SYSOPR 用户的权限不记录在 DB2 的 Catalog 表中，同时此用户在执行操作的时候，不需要 DB2 Catalog 表进行权限验证。
- Installation SYSOPR 用户的权限不可使用 REVOKE 命令取消，只能通过修改 DSNZPARM 中的相关参数，并重新编译 DSNZPARM 后才可生效。

【DBMAINT】

DBMAINT 管理权限是针对 DATABASE 的，用户一旦被授予了 DBMAINT 权限后，可以执行如下操作：

- 在 DATABASE 里创建对象。
- 运行不会更改数据的 Utility。
- 执行 DB2 命令。
- 可以终止除 DIAGNOSE, REPORT 和 STOSPACE 以外的 Utility。

【DBCTRL】

DBCTRL 管理权限除了拥有 DBMAINT 所拥有的权限外，还可以运行能够更改数据的 Utility。

【DBADM】

DBADM 管理权限是除了拥有 DBCTRL 所拥有的权限外，还拥有如下权限：

- 可以通过 SQL 语句访问数据表中的记录。
- 可以删除或者修改数据库中的表空间、表和索引。
- 对数据库中的表使用 COMMENT、LABEL 或 LOCK TABLE 语句。
- 对数据库中的索引使用 COMMENT 语句。

【SYSCTRL】

SYSCTRL 管理权限除了不可访问应用数据外，拥有绝大部分数据库的维护及管理权限：

- 作为 Installation SYSCTRL 或 DBCTRL。
- 运行任意的 Utility。
- 对数据库中的表使用 COMMENT、LABEL 或 LOCK TABLE 语句。
- 可以基于任意 CATALOG 表创建 VIEW。
- 可以 BIND 任意 PLAN 和 PACKAGE。

【SYSADM】

SYSADM 管理权限包含了 SYSCTRL，PACKADM 和 DBADM 的所有权限，可以访问所有的应用数据，除此以外还拥有如下特权：

- 可以对任意 PLAN 和 PACKAGE 执行 EXECUTE 和 BIND 操作。
- 将 SQLID 设置为任意 ID。
- 可以在其他用户创建的表上创建或删除 synonyms 和 views。
- 删除 DSNDB07 数据库。

【Installation SYSADM】

在 DB2 子系统进行搭建的时候，可以指定 1～2 个 ID 作为 Installation SYSADM，在拥有 SYSADM 所有权限的同时，还拥有如下特权：

- Installation SYSADM 用户的权限不记录在 DB2 的 Catalog 表中，同时此用户在执行操作的时候，不需要 DB2 Catalog 表进行权限验证。
- Installation SYSADM 用户的权限不可使用 REVOKE 命令取消，只能通过修改 DSNZPARM 中的相关参数，并重新编译 DSNZPARM 后才可生效。

2.8.3.4　如何获得管理权限

数据库管理员需要通过 SQL DCL 的 GRANT 语句对特定用户 ID 授予管理权限，也可以使用 DCL 的 REVOKE 回收权限。

2.8.3.5　隐性权限

隐性权限就是指对象的所有者对它所创建的对象拥有一系列权限，表 2.15 列出了 DB2 中各类对象的拥有者所带的隐性权限。

表 2.15　隐性权限一览表

对象类型	对象拥有者的隐性权限
Alias	To drop the alias
Database	DBCTRL or DBADM authority over the database, depending on the privilege (CREATEDBC or CREATEDBA) that is used to create it. DBCTRL authority does not include the privilege to access data in tables in the database
Distinct Type	To use or drop a distinct type
Index	To alter, comment on, or drop the index
JAR (Java class for a routine)	To replace, use, or drop the JAR

续表

对象类型	对象拥有者的隐性权限
Package	To bind, rebind, free, copy, execute, or drop the package
Plan	To bind, rebind, free, or execute the plan
Sequence	To alter, comment on, use, or drop the sequence
Storage Group	To alter or drop the group and to name it in the USING clause of a CREATE INDEX or CREATE TABLESPACE statement
Stored procedure	To execute, alter, drop, start, stop, or display a stored procedure
Synonym	To use or drop the synonym
Table	To alter or drop the table or any indexes on it To lock the table, comment on it, or label it To create an index or view for the table To select or update any row or column To insert or delete any row To use the LOAD Utility for the table To define referential constraints on any table or set of columns To create a trigger on the table
Tablespace	To alter or drop the table space and to name it in the IN clause of a CREATE TABLE statement
User-defined functions	To execute, alter, drop, start, stop, or display a user-defined function
View	To drop, comment on, or label the view, or to select any row or column ； To update any row or column, insert or delete anyrow (if the view is not read-only)

2.8.3.6 隐性权限授权方法

如果一个用户是某一个 DB2 对象的拥有者，那么他可以对任何人授予对此对象进行操作的各类权限。例如一张表的拥有者可以对任何人授予对此表进行读取的权限：G RANT SELECT ON TABLE3 TO USER4

2.8.4 DB2 审计

本节主要介绍对 DB2 进行审计的一些知识，范围涉及审计的目的、审计的内容、和几种具体的审计方法。学习完本节读者将初步掌握 DB2 审计的方法。

2.8.4.1　DB2 审计的目的

DB2 系统是一个企业业务的核心，数据信息是企业最重要的资源，需要受到严格的保护。有效的 DB2 审计能够针对用户和对象的敏感操作进行审计，对误操作或违规活动及时进行事后检查，并生成有价值的报告。

对 DB2 数据库进行定期事后监督检查，对维护数据库系统和数据库信息的安全性是至关重要的。

2.8.4.2　DB2 审计的内容

审计内容主要包括特定用户在特定时间内的如下操作：

- 对数据库对象的创建、修改、删除等操作。
- 包括对数据库、表空间、表、索引、视图等对象的创建、修改、删除操作。
- 对 DB2 表的插入、更新、删除等操作。
- 包括对 DB2 表内数据的插入、更新、删除等操作。
- 授权成功、失败、权限发生变化，以及因权限不足而操作失败。
- 包括成功或失败的授权操作以及其他能够引起权限变化的操作。
- DB2 命令操作。
- 包括启动、停止和改变数据库对象状态的操作。
- 绑定操作。
- 包括对程序计划和包的绑定操作。
- DB2 工具程序操作，包括可能涉及更改 DB2 数据内容的工具程序，例如装载、重组、检查数据完整性及一致性的操作。

2.8.4.3　DB2 审计的方法

1. 使用 DB2 CATALOG

DB2 CATALOG 中有一类表，记录 DB2 内部授权信息，如表 2.16 所示。在使用 DB2 CATALOG 表进行审计的时候，可以通过选择相应的 DB2 CATALOG 表来查看各类用户对不同数据对象权限，检查授权的合理性。

表 2.16　授权信息表

DB2 CATALOG 表名	记录内容
SYSIBM. SYSCOLAUTH	DB2 表中各字段的授权记录

续表

DB2 CATALOG 表名	记录内容
SYSIBM. SYSDBAUTH	DB2 中各 Database 的授权记录
SYSIBM. SYSPLANAUTH	DB2 中各 Plans 的授权记录
SYSIBM. SYSPACKAUTH	DB2 中各 Packages 的授权记录
SYSIBM. SYSRESAUTH	DB2 中各类资源的授权记录(BP,SG 等)
SYSIBM.SYSROUTINEAUTH	DB2 中各存储过程的授权记录
SYSIBM.SYSSCHEMAAUTH	DB2 中各 Schemas 的授权记录
SYSIBM.SYSTABAUTH	DB2 中各表和视图的授权记录
SYSIBM.SYSUSERAUTH	DB2 中各用户及组的授权记录

2. 使用 DB2 Utility: DSN1LOGP 进行日志审计

DSN1LOGP 是一个 DB2 Stand Alone Utility，执行此 Utility 的时候，通过指定一段 DB2 LOG RBA 值的范围，可以打印出这一段范围 DB2 内发生的动作，用于判断操作是否违规。但是缺点是打印出的结果可读性较差，需要数据库管理员对结果进一步分析格式化。此 Utility 的具体语法和参数请参考 4.2.3 节。

3. 使用 DB2 日志分析工具 LOG ANALYSIS 进行审计

在对 DB2 进行审计的时候，我们还可以借助工具软件进行。LOG ANALYSIS 就是这样一个工具，使用时通过指定一段时间的日志进行打印，可以获得数据库系统在这一段时间内的相关操作，例如对数据库表的插入、删除及修改等操作。该工具提供的报告可读性较强。

4. 用 DB2 AUDIT TRACE 进行审计

使用 DB2 AUDIT TRACE 进行安全审计的时候，主要是通过打开 DB2 审计相关 TRACE，将需要审计的信息收集到特定的 DB2 日志当中，然后通过打印报告来进行相应的安全审计。

DB2 AUDIT TRACE 主要有 8 种类型，表 2.17 描述了各类 TRACE 所收集的信息内容。

表 2.17 trace 信息表

	CL	收集信息的类型	IFCID
AUDIT TRACE	1	授权失败相关信息	140
	2	授权（GRANT）和取消授权（REVOKE）	141
	3	针对审计的表进行的 CREATE，ALTER 和 DROP 操作	105, 107, 142
	4	针对审计的 OBJECT 的首次修改	105, 107, 143

续表

	CL	收集信息的类型	IFCID
AUDIT TRACE	5	针对审计的 OBJECT 的首次读取	105, 107, 144
	6	BIND 时发生的 SQL 语句	105, 107, 145
	7	针对审计的 OBJECT 发生的授权变化	55, 83, 87, 169, 319
	8	使用 Utility 对审计的 OBJECT 的访问	24, 105, 107

通过打开以上类型的 DB2 AUDIT TRACE，可以获取许多有助于安全审计的信息，主要有以下几种：

（1）发生权限失败的所有实例。如 USER1 试图从一个未授权的表中 SELECT 信息。

（2）GRANT REVOKE 语句的执行。

（3）通过指定 AUDIT CHANGES 或 AUDIT ALL 创建的特定表的每个 DDL 语句。

（4）审核表的第一个 DELETE、INSERT、UPDATE 语句（需指定 AUDIT CHANGES 属性）。

（5）只指定 AUDIT ALL 创建的表的第一个 SELECT 语句。

（6）审计表的 BIND。

（7）执行 SET CURRENT SQLID 语句后产生的所有权限 ID 更改。

（8）DB2 实用程序的执行情况。

下面是对某张数据表使用 AUDIT TRACE 审计的步骤，供读者参考。

STEP 01 根据审计对象的不同，制定审计的策略。

STEP 02 需要审计某张表，修改该表的属性为 AUDIT CHANGE 或 AUDIT ALL。

STEP 03 打开相应的 AUDIT TRACE 。

STEP 04 收取 Trace 数据。

STEP 05 停止 AUDIT TRACE。

STEP 06 使用 DB2 PM 生成格式化的审计报表。

对 DB2 进行安全审计虽然可以在很大程度上保证数据库信息的安全性，但是也会增加系统的各类开销，例如启动 DB2 AUDIT TRACE 的时候，每个 TASK 将增加 5%的 CPU 消耗，同时也会增加系统存储等资源的消耗，在一定程度上影响系统性能。因此数据库管理员应该根据应用对安全性的要求，灵活地打开或关闭审计功能。

建议：只启动 TRACE1、2、7，用于审核权限失败、改变和实用程序，除了这些类型的处理，不会增加其他负载。因为大多数 TASK 不会导致权限失败和发布 GRANT REVOKE

或实用程序。

2.9 课后习题

1．如下 DB2 系统的地址空间，哪个不是必需的（　　）。

A．ssnmMSTR　　B．ssnmIRLM

C．ssnmDBM1　　D．ssnmDIST

2．如下 DB2 系统的地址空间，记录日志是哪个地址空间的职责（　　）。

A．ssnmMSTR　　B．ssnmIRLM

C．ssnmDBM1　　D．ssnmDIST

3．DB2 系统需要支持哪种标准，才可以接受来自其他 DB2 系统的远程访问（　　）。

A．DRDA　　B．SNA

C．TCP/IP

4．一个更新的锁在使用可重复读的隔离级别的应用程序时，在以下哪种情况下被释放（　　）。

A．访问该行的游标关闭时

B．当应用程序提交 ROLLBACK 时

C．当存取该行的游标移到下一行时

D．当应用程序的更新通过 UPDATE 语句实现时

5．以下哪种数据库对象不会获取锁（　　）。

A．一行　　B．一个表

C．一列　　D．一个索引的键值

6．对于一个只读的数据库，应用程序选择何种隔离方式能够达到最大并发读（　　）。

A．读稳定性　　B．可重复读

C．游标稳定性　　D．未提交读

7．请判断如下一句话是否正确：DBRM 是应用程序在预编译后产生的，对 DBRM 进行 BIND 后，应用程序才可以访问 DB2 的数据。

8．如下哪种谓词的访问效率最高（　　）。

A．matching　　B．screening

C．stage 2

9．请判断下面的话是否正确？stage 1 的谓词好于 stage 2 的谓词。

10．在下面的 SQL 中，P1、P2 和 P3 都是 simple 谓词：

```
SELECT * FROM T1 WHERE P1 AND (P2 OR P3);
```

不是 Boolean Term 谓词的是（　　）

A．P1　　B．P2 OR P3

C．P1 AND (P2 OR P3)　　D．P2　　E．P3

11．假设在表 T 中的列（C1，C2，C3，C4）上存在一个索引，并且每个列中 0 是最小的值。请问 SELECT * FROM T WHERE C1=5 AND C2=7 的访问路径是怎样的（　　）。

A．table space scan　　B．index only

C．maching index scan

12．请判断下面的话是否正确？RUNSTATS 不会影响访问路径的选择。

13．假设表 T 的列 C1 仅仅包含 5 个不同的值：A、D、Q、W 和 X。在不知道其他信息的情况下，DB2 估计五分之一的记录在列 C1 上拥有值 D。那么，对于表 T 来说，谓词 C1=‘D’的过滤率为（　　）。

A．0.1　　B．0.2

C．0.3　　D．0.4

14．如下 JOIN 操作中，效率最高的 JOIN 操作是（　　）。

A．full outer join　　B．left outer join

C．right outer join　　D．inner join

15．如果动态 SQL 的效率不高，BIND 时可以使用如下的哪个参数来优化这个 SQL 每次运行时的效率（　　）。

A．REOPT(VARS)　　B．REOPT(ONCE)

C．REOPT(NONE)

16．请简述 ID 的类型。

17．请简述显性权限。

18．请简述隐性权限。

第 3 章　DB2 Data Sharing基础

本章主要介绍DB2 Data Sharing方面的知识，内容包括Data Sharing特性介绍、与 Stand Alone 的区别、Data Sharing 与 SYSPLEX 的关系、Data Sharing 系统架构、Data Sharing 的实现和恢复等，帮助读者掌握 Data Sharing 的基本概念和基本使用方法。

3.1　DB2 Data Sharing 介绍

随着 DB2 产品在越来越多的企业和领域的应用，当业务发展越来越快、需要处理的数据请求越来越多时，DB2 处理的数据量也相应增加，扩充容量可能随时发生。单一的 DB2 子系统往往不能满足成倍增长的数据请求。另外，多个 DB2 子系统共享数据的需求也越来越普遍，在 DB2 V4 版本之前，在多个子系统之间共享 DB2 数据的唯一方法是使用分布的 DB2 连接方式，这种方式存在一定的缺点。IBM 公司引入了数据共享组的系统架构（DB2 Data Sharing Group），DB2 数据共享允许运行在多个 DB2 子系统上的应用程序并发地读/写相同的数据集，简单地说，数据共享使多个 DB2 子系统能像一个系统那样运行。

本节着重介绍 Data Sharing 的基础知识，主要包括 Data Sharing 依托的系统架构——Parallel Sysplex 的介绍、DB2 Data Sharing 架构的优势。

3.1.1　Parallel Sysplex与DB2 Data Sharing简介

Data Sharing 技术允许多个应用程序并行读写同一份 DB2 应用数据。应用程序可以运行在不同的 DB2 子系统中，而运行 DB2 Data Sharing 架构需要耦合系统（Parallel Sysplex）架构的支持。

Sysplex 是 Systems Complex 的缩写，是由一组能够相互通信的 IBM z/OS 主机构成的。它们之间通过时钟控制器（Sysplex Timer）手段来进行时钟同步。Parallel Sysplex 则是 Sysplex 的一种，它采用 CFs（Coupling Facilities）耦合技术，为所有在 Sysplex 系统中的应用程序提供更强的数据缓存、数据处理、数据锁管理的能力。

图 3.1 描述的就是 Parallel Sysplex 系统，它将不同的主机系统通过 CF 耦合实体、时钟控制器设备和光纤通道整合到一起，构成一个功能更加强大的综合处理实体。

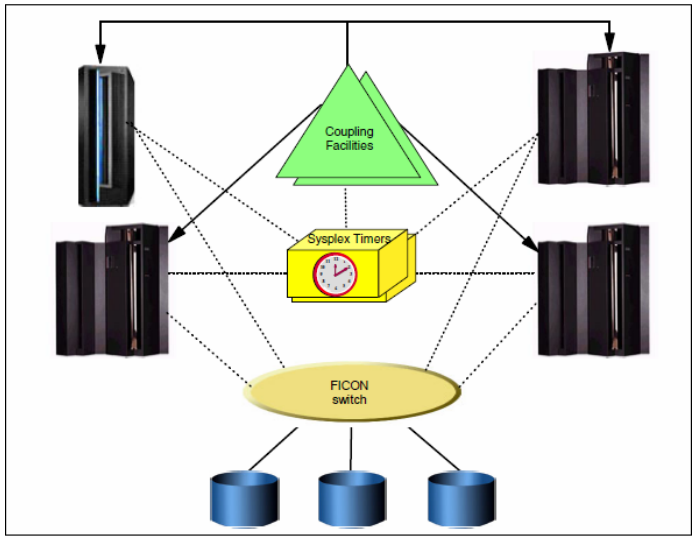


图 3.1　Parallel Sysplex 系统

图 3.1 中各个主机之间通过 Coupling Facilities（CFs）互联通信，并且通过时钟控制器保持时钟同步，DB2 数据库可以安装在一个或多个主机系统上。

在 Parallel Sysplex 系统中，多个 DB2 子系统连接起来构成一组 DB2 Data Sharing Group，这些 DB2 子系统访问共同的一份应用数据。我们把同属于 DB2 Data Sharing Group 的每个 DB2 子系统称作 DB2 共享组的一个成员（MEMBER）。

每一个 DB2 Data Sharing Group 中的 MEMBER 必须属于同一个唯一的 DB2 Data Sharing Group，DB2 Data Sharing Group 中的所有 MEMBER 共享一份相同的 DB2 日志（Log）和系统控制表，属于同一个 DB2 Data Sharing Group 的 MEMBER 必须在同一个 Parallex Sysplex 中。到目前为止，DB2 Data Sharing Group 所允许的最大 MEMBER 个数为 32。

图 3.2 显示了 DB2 Data Sharing Group 中的 MEMBER 情况，DB2 Data Sharing Group 中的任意成员可以访问 DB2 Data Sharing Group 中的同一份共享数据。处于 DB2 Data Sharing Group 组中的 DB2 子系统可以在同一时刻读取和更新这些数据。

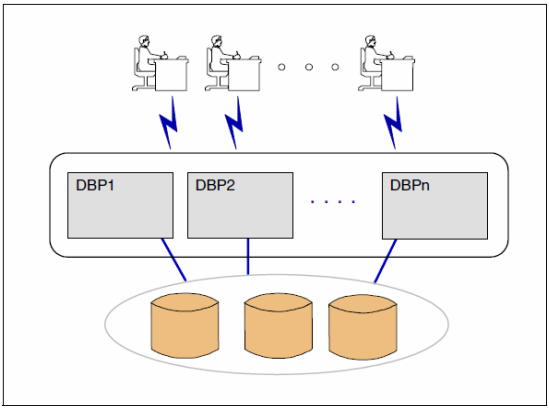


图 3.2　DB2 Data Sharing Group

在没有 Data Sharing 的环境中，各个 DB2 子系统只能访问独立的数据。如图 3.3 所示，DB2 系统间是相互独立运行的。当某个 DB2 不可用时，用户不能访问相关数据库信息及其使用数据。

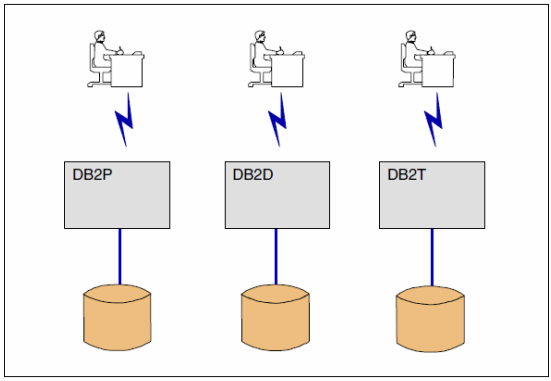


图 3.3　独立的 DB2 系统

3.1.2　DB2 Data Sharing技术的优势

Data Sharing 技术给客户带来高度的可扩展性、高效的性能及数据的连续可用性。通过采用 DB2 Data Sharing 的架构可以给用户带来以下好处：

- 提高 DB2 数据的可用性；
- 提高 DB2 系统的处理能力；
- 更加灵活的 DB2 系统环境配置；
- 提升交易处理能力。

接下来将具体讲解采用 Data Sharing 技术在各个方面所带来的变化。

3.1.2.1　提高数据可用性

数据信息在任意时间和条件下都能够被访问是保证客户业务连续性的前提。随着业务数据信息量的激增、信息系统架构的复杂程度增加，以及软硬件设施自身的故障，数据信息的出错和不可访问的情况在所难免，如何有效降低业务被中断的时间已经成为各个用户考虑的重点。使用 DB2 Data Sharing 技术可以极大地提高系统可用性，实现用户在任何时候都可以访问所需的数据。

Data Sharing 环境下，用户可以通过多个 DB2 MEMBER 来访问数据，这意味着当 Data Sharing Group 中的一个 MEMBER 发生故障时，应用程序仍然可以通过其他 MEMBER 访问到数据库相关数据。如图 3.4 所示，当 DBP1 发生故障时，共享管理机制可以动态地将用户访问数据的需求传递给其他数据库 MEMBER，然后将相关数据返回给用户。只要多条路径中有一条是可以正常使用的，那么用户的应用程序就可以实现对数据的正常访问。

增加数据的可用性会在一定程度上带来性能的开销，因为在 DB2 Data Sharing Group 中的 DB2 子系统 MEMBER 越多，那么它们之间的通信交互开销就越多。目前通过 CF 耦合体和 Parallel Sysplex 提供的高效锁机制、高效的内存管理机制可以使其开销尽可能地小。

3.1.2.2　可扩展性的提高

当业务发展越来越快、需要处理的数据请求越来越多时，DB2 处理的数据量也相应增加，扩充容量可能随时发生。单一的 DB2 子系统往往不能满足成倍增长的数据请求。下面

将对比没有采用 Data Sharing 技术和采用了此技术在扩展性上的变化。

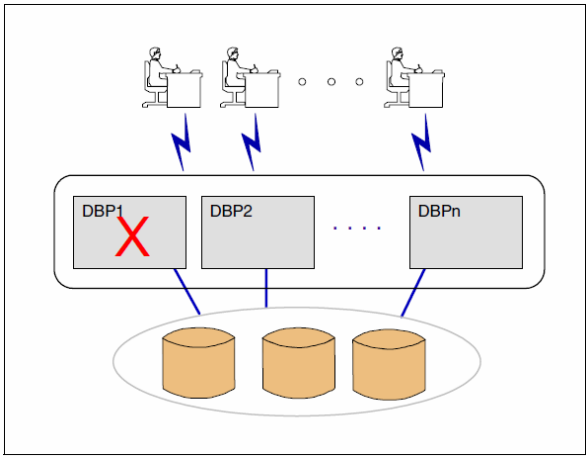


图 3.4　DB2 Data Sharing Group 的高可用性

1. 没有使用 DB2 Data Sharing 技术

当没有使用 Data Sharing 技术的时候，用户只能采用如下的方式来应对容量和业务需求的增加。

（1）将数据复制到或是拆分到额外的、独立的 DB2 子系统中。

此方法需要用户维护多套独立的 DB2 子系统中的数据，由于这些 DB2 子系统间并没有建立相互的连接，因此维护起来很不方便。

（2）重新配置安装一个性能更好、容量更大的 DB2 子系统，然后将旧的数据和新的数据一并保存到新的 DB2 子系统中去。

这种方法需要用户对业务的发展有很好的判断力，否则不断更新系统，成本会很高，并且此种方法对废弃的旧系统来说也是一种资源的浪费。

2. 使用 DB2 Data Sharing 技术

（1）可以支持业务的动态增长需求。

Parallel Sysplex 允许用户在不间断业务的情况下动态增加软硬件来提升业务处理能力，它将多个 DB2 子系统结合起来组成 Data Sharing Group，其中各个 DB2 子系统可以访问共同的数据，这给 DB2 管理员的维护工作带来了更多的便利。

（2）增加负载均衡性。

Data Sharing 提供的均衡负载的能力使得当用户动态增加一个新的 DB2 MEMBER 到

Data Sharing Group 后，不必考虑如何分配多个 DB2 MEMBER 间的负载，Data Sharing 会自动将负载动态分配到各个 DB2 MEMBER 上，以达到整个 Data Sharing Group 均衡负载的效果。

图 3.5 描述了 Data Sharing Group 中的 MEMBER 如何在不间断业务的情况下增加 DB2 子系统个数。Transaction Managers 组件将用户传来的处理请求，均衡地分散到多个 DB2 子系统上，使得每个 DB2 子系统的负载都相对保持平衡。图中的各个层次间独立、透明，每个层次上的容量变化的相关操作都不会影响其他层次，因而不会对业务的连续性产生影响。

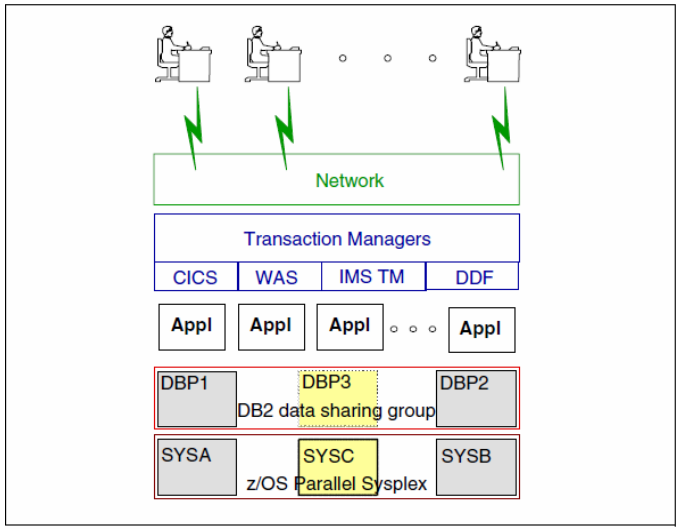


图 3.5　独立的层次关系和容量的灵活扩充能力

3.1.2.3　灵活的配置管理能力

Data Sharing 技术使得 DB2 系统环境的配置更加灵活，当具体使用的需求发生变化时，管理员无须做出复杂的修改操作。举例说明如下。

（1）在同一个 Parallel Sysplex 上可以构建多个 DB2 Data Sharing Group，用户可以对不同的 Data Sharing Group 设定不同的工作需求，例如有的用于测试，有的用于生产，并且根据不同的工作需求来配置不同的参数。

（2）可以根据实际业务处理需求灵活调整 DB2 Data Sharing Group 中启动的 DB2 MEMBER 数量，从而灵活控制整个组的性能容量。

在系统管理层面，DB2 Data Sharing 使得对多个数据库在系统中的统一管理更为便捷。

3.1.2.4 更强的处理能力

Data Sharing 技术可以加强业务的处理能力。如图 3.6 所示，终端用户的数量较图 3.3 有所增加，表示实际的处理能力的增强。因为在 DB2 Data Sharing Group 情况下，多个 DB2 子系统上能够运行同一个应用程序，这样单位时间的处理能力显著增强，提供给终端用户的业务处理量也就相应增多了。

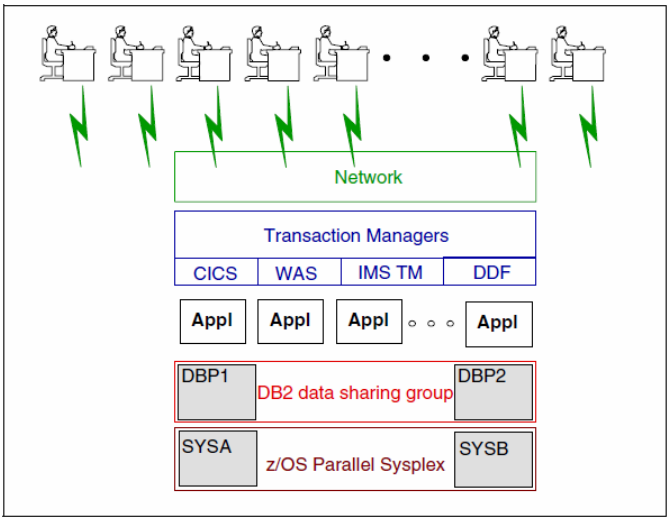


图 3.6 业务处理能力增加

3.1.2.5 保持原有的应用程序不变

使用 Data Sharing 技术，并不需要用户修改原有的应用程序，所有在非 Data Sharing 环境下使用的程序都能继续使用。

3.2 DB2 Data Sharing 体系架构

DB2 Data Sharing 是基于共享一份数据的一种体系架构，这种同一个 DB2 Data Sharing Group 中的多个 MEMBER 可以访问同一份共享数据的架构，给用户带来很强的可扩展性。另外，这种体系架构配置灵活、管理方便，能够为用户提供良好的负载均衡性。本节将重点介绍 Data Sharing 的系统架构。

3.2.1　DB2 Data Sharing架构的问题及解决方法

在 DB2 Data Sharing 环境下，由于多个 DB2 MEMBER 会同时访问同一份应用数据和系统数据，这会带来以下问题：

- 多个 DB2 MEMBER 之间如何进行通信。
- 多个 DB2 MEMBER 如何处理锁。
- 多个 DB2 MEMBER 同时访问数据，如何保证数据的一致性。

问题的解决途径是引入 CF（COUPLE FACILITY），通过在 CF 中定义三类 Structure（LIST Structure、LOCK Structure、CACHE Structure）来解决以上的问题。

CF 定义的示意图如图 3.7 所示。

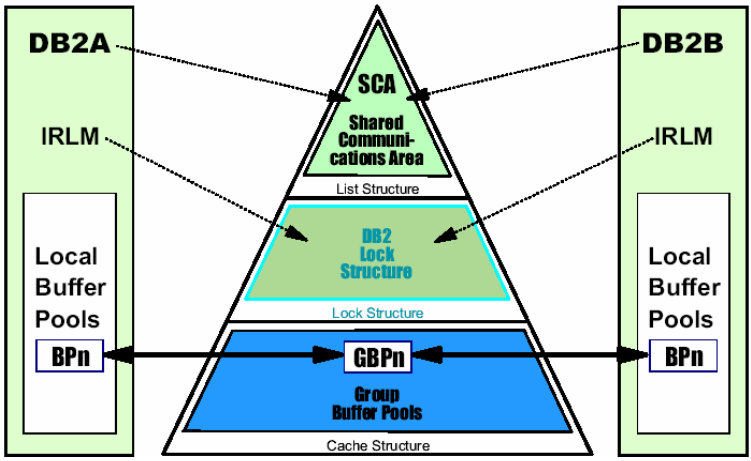


图 3.7　CF 定义的示意图

1. List Structure——解决通信问题

List Structure（SCA-Shared Communication Area），又称作 SCA，存放了整个 Group 的一些系统信息，用来确保 Data Sharing 的正常运行。每个 DB2 MEMBER 都通过 SCA 向其他 DB2 MEMBER 传递控制信息，如：数据库的异常状态信息、关于每一个 MEMBER 的 Logs 的信息、每一个 MEMBER 的系统信息等。

2. Lock Structure——解决锁问题

DB2 通过 Lock Structure 来实施 Global Locking 的功能。Lock Structure 可以保护共享的应用数据（包括表或数据页等），从而保证多个 DB2 MEMBER 并发访问共享的应用数据。

同时，DB2 借助 SYSPLEX 的 XES（Extended Services）中的 SLM（System Lock Manager）负责将每个 DB2 MEMBER 的 lock 信息传递给 CF 中的 Lock Structure。

3. Cache Structure——确保数据一致

Cache Structure（Group Buffer Pools），又称作 GBP，对 DB2 MEMBER 提供共享应用数据的缓存功能。DB2 通过它来确保数据的一致性。

GBP 需要与 LBP（Local Buffer Pool）一起使用。整个 Data Sharing Group 中只有 1 套 GBP，存储在 CF 的 structure 中。Data Sharing Group 中的每个 MEMBER 各有 1 套 LBP，存储在 DB2 DBM1 地址空间的内存中。每个 MEMBER 的 LBP 个数和命名必须一致（bp0-bpN）。

LBP 必须与 GBP 的个数一致，保证一一对应关系，如：bp0-gbp0，bpN-gbpN。

下面的章节将详细讲解并发性和一致性的控制原理。

3.2.2 并发性和数据一致性控制

在 DB2 Data Sharing 环境中，多个 DB2 MEMBER 会同时访问一套共享的应用数据，在这种情况下，如何保证并发性和数据一致性至关重要。DB2 Data Sharing 通过并发访问控制（Global Locking，全局锁机制的控制）来保证并发性，通过对修改过的数据的一致性控制（Managing Changed Data）来控制数据一致性。

在 Data Sharing 环境中，DB2 通过全局锁机制的控制来实现对并发访问的控制。举例说明：如果有两个程序同时要读取同一张表的某条数据，DB2 的全局锁机制就会允许对于这些读取操作同时进行；如果有两个程序同时要访问同一张表的某条数据，但是一个程序要读取数据，另一个程序要修改数据，那么 DB2 的全局锁机制就会让这两个程序串行执行，要么是先读，要么是先修改。

在 Data Sharing 环境中，DB2 会将修改过的数据放到 CF 中的一个全局缓存区（Group Buffer Pool）内，如果多个 DB2 MEMBER 需要访问这些数据，都从这个缓存区中读取，从而保证访问数据的正确性。

DB2 Data Sharing 要实现上述两个功能，必然会增加一些额外的 CPU 和其他一些开销，为此建议以下两点：

- 尽可能通过人为方法来控制并发访问。例如，某些程序都是对数据的读取操作，而某些程序都是对数据的修改操作，如果能够实现人为控制这两类操作串行进行，将

大大减少 DB2 开销；

- 尽可能使用性能更好的 CPU 和 CF，使用更快的光纤通道，来实现更快的传输速度，实现更快的数据访问。

3.2.2.1　全局锁机制（Global Locking）和CF中的Lock Structure

DB2 通过内部资源锁管理器（Internal Resource Lock Manager，IRLM）来管理 DB2 锁。在 DB2 Data Sharing 环境中，每个 DB2 MEMBER 都有一个各自的 IRLM，来管理其内部的锁。在 Parallel Sysplex 环境下，IRLM 通过使用 z/OS 和 CF 之间的通信组件 XES（cross-system Extended Services）来实现多个 DB2 MEMBER 之间锁的传递，从而实现上面提到的全局锁机制的功能。

在 DB2 Data Sharing 环境中，在 CF 的 Lock Structure 中会维护一张哈希表，此表中记录着某个正被访问的 DB2 资源 A 及其相关锁信息，当某个 IRLM 需要申请访问某个 DB2 资源的全局锁（Global Lock）的时候，那么此请求会通过 XES 以很快的速度传递到 CF 上（此传递速度是毫秒级别的），CF 会访问并更新这张哈希表中的相关记录，成功之后将结果返回给 IRLM。

此时，如果有另外一个 DB2 MEMBER 需要访问 DB2 资源 B，并向 CF 申请另一个全局锁，CF 会在此锁的申请成功处理后，将结果返回给申请锁的 IRLM（如图 3.8 所示）。此时，这两个 DB2 IRLM 之间并没有信息传递。

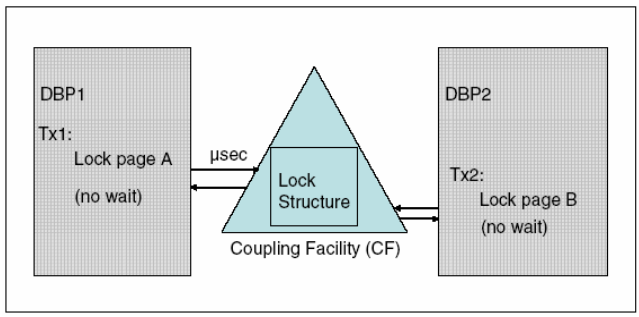


图 3.8　Data Sharing 环境下的全局锁申请

但是，如果两个 DB2 IRLM 都向 CF 申请相同的 DB2 资源的全局锁，就会产生冲突，或者称为锁之间的竞争（Lock Contention）。此时 CF 会将此 Lock Contention 信息反馈给 IRLM，之后两个 IRLM 之间会传递一些信息，来让其中一个 DB2 IRLM 等待，以解决此 Lock Contention。

在 DB2 Data Sharing 环境中，有两种类型的锁，逻辑锁（Logical Lock，简称 L-LOCK）和物理锁（Physical Lock，简称 P-LOCK）。L-LOCK 是传统意义上的锁，在 Data Sharing 和非 Data Sharing 环境中，DB2 都使用 L-LOCK 来控制访问的并发性。P-LOCK 只在 Data Sharing 中才有，DB2 从磁盘上读取某个文件，并将相关数据写入缓存区的时候，会有 P-LOCK，其目的是保证缓存区中的数据的一致性。

如图 3.9 所示，DB2A 对 P1 持有 S 锁，这是 DB2B 想申请对 P1 的 X 锁，但是通过向 CF 的 Lock Structure 申请时发现 X 锁是和其他任何锁都不兼容的，出现锁资源的冲突，只能等待，直到 DB2A 释放对 P1 的锁后，DB2B 才能取得 P1 的锁，对于 DB2A 的程序 TX3 申请对 P1 的 S 锁，通过对 CF 的 Lock Structure 的检查，发现 S 锁是和 X 锁兼容的，因此可以访问 P1，无须等待。

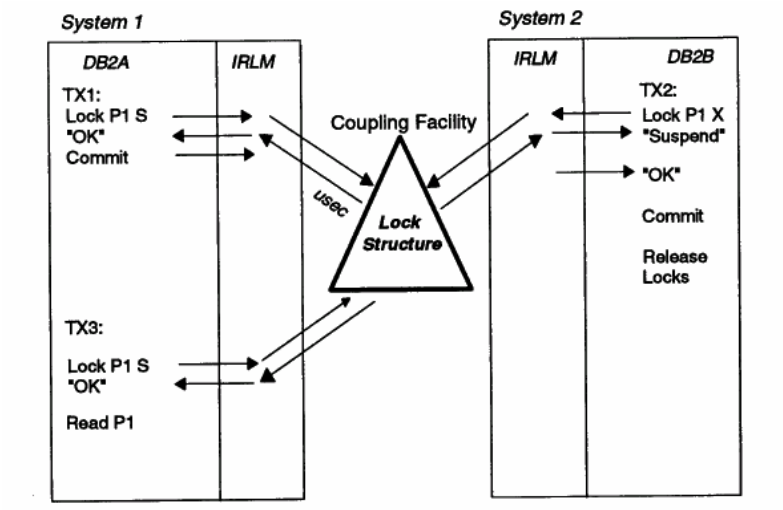


图 3.9　锁冲突示意图

锁的冲突（Lock Contention）通常发生在多个 MEMBER 需要在同时使用同一个资源时。在 DB2 Data Sharing group 环境中有三类冲突。

- **Real contention**：发生在锁的请求不兼容的情况下，如一个程序持有 S 锁，而另一个程序想要对同一个页面申请 X 锁。
- **False contentions**：发生在由于哈希算法导致对于不同资源映射到同一个哈希值的情况。
- **XES contentions**：发生的情况是由于 XES 只能识别两种锁类型（S 和 X），而 IRLM

可以支持更多的锁类型（S,X,U,IX…），这样从 IRLM 角度上看是兼容的锁上传到 MVS 的 XES 时会存在被认为是不兼容的锁冲突的情况。

当一个锁冲突发生时，请求方的执行语句只能等待直到冲突被解决。如果冲突是真实的，请求方会等待直到不兼容的锁被释放或程序因为超时或死锁而异常中断。

3.2.2.2　对修改数据的管理和全局缓存区（Group Buffer Pool）

为了提高数据访问效率，减少 CPU 和 IO 的开销，DB2 会将访问过的数据放在本地缓存区（Local Buffer Pool）中，以备下次使用。在 Data Sharing 中，每个 DB2 MEMBER 都有自己的本地缓存区。如何保证这些 Local Buffer Pool 中保存的数据的一致性，对于 DB2 Data Sharing 至关重要。

例如，在某个 Data Sharing 环境中，两个 DB2 MEMBER 都访问过同一份数据，那么这份数据会保留在这两个 DB2 MEMBER 的 Local Buffer Pool 中。此时，其中一个 DB2 MEMBER 中的程序修改了这份数据，那么另一个 DB2 MEMBER 的 Local Buffer Pool 中的数据就不是最新的了，如何保证在两个 DB2 MEMBER 的 Local Buffer Pool 中的数据一致，对于 DB2 Data Sharing 至关重要。

DB2 Data Sharing 通过存放在 CF 中的全局缓存区（Group Buffer Pool），来保证在多个 DB2 MEMBER 的 Local Buffer Pool 中的数据一致性。

其原理说明如下：如果某个 DB2 Data Sharing 中的多个 DB2 MEMBER 都要访问同一份数据，那么 DB2 会将这份数据放到各自的 Local Buffer Pool 中。之后如果某个 DB2 MEMBER 中的程序要修改这份数据，那么 DB2 会在这份数据中增加一个“GBP-dependent”标志，其含义是在数据修改完毕后，还需要同步更新 Group Buffer Pool 中的数据。在 CF 中的 Group Buffer Pool 中的数据被更新后，CF 会通知其他的 DB2 MEMBER，这份数据已经被更新了。DB2 MEMBER 收到此信息后，会将各自的 Local Buffer Pool 中保存的原数据置为“不可用（Invalidate）”状态。之后再有程序通过 DB2 访问这份“不可用”数据的时候，DB2 会自动从 CF 中下载此数据的最新版到其 Local Buffer Pool 中，从而保证了多个 DB2 MEMBER 的 Local Buffer Pool 中的数据一致性。

DB2 的 Group Buffer Pool 由 Directory entries（目录）和 Data elements（数据元素）两部分组成。Directory entries 的作用是跟踪记录“GBP-dependent”的数据页（Page）访问信息。当某个 DB2 MEMBER 需要访问“GBP-dependent”的 Page 时，就会将此 DB2 MEMBER 的访问信息记录到 Group Buffer Pool 的某个 Directory entry 中。一个 Directory entry 可以记

录 Data Sharing 中所有 DB2 MEMBER 对特定的“GBP-dependent”的 Page 的访问情况。Data elements 的作用是保存修改后的最新的“GBP-dependent”的 Page，相当于缓存的作用，从而使 Data Sharing 中的所有 DB2 MEMBER 都可以高效访问这些数据。

经过一段时间，DB2 就需要将保存在 Group Buffer Pool 中的数据页写到磁盘上的数据库表文件中，这个过程叫做“Castout”。由于 CF 中的 Group Buffer Pool 并没有和磁盘直接相连，“Castout”的过程是通过 Data Sharing 中某个 DB2 MEMBER 协助完成的，具体过程是 DB2 先将 Group Buffer Pool 中的数据页传递到某个 DB2 MEMBER 的地址空间的内存中，再从内存写到磁盘文件中，这个 MEMBER 称为“Castout Owner”。在 Data Sharing 中，每个 DB2 MEMBER 都可以称为“Castout Owner”。

图 3.10 介绍了在两个 MEMBER 的 DB2 Data Sharing 中，DB2 MEMBER DBP1 正在执行“Castout”动作。

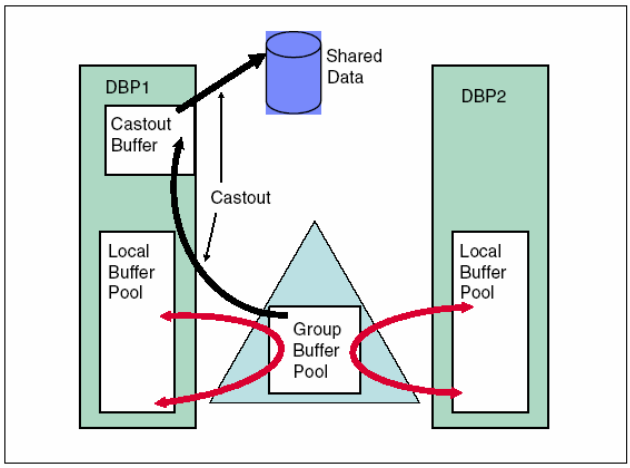


图 3.10　Castout 的处理过程

当某个数据页信息被写入 GBP 的 Data element 中的时候，相应的 Directory entry 中一定也记录了 Data Sharing 中的所有访问这个数据页的 DB2 MEMBER 信息。此时，CF 就会通过 Coupling Facility Control Code（CFCC）向 Directory entry 记录的所有 DB2 MEMBER 发送一条信息，告知此数据页已经被更新了，最新数据放在 GBP 中。在 DB2 MEMBER 收到这条信息后，会将各自的 Local Buffer Pool 中的数据页置为“不可用（Invalidate）”状态。这个过程叫做“Cross-invalidation”。

当某个数据库 MEMBER 中的程序需要访问这些“不可用（Invalidate）”的状态的数据

页时，就会从 GBP 中读取相关数据页来访问。如果这些数据页已经不在 GBP 中了，那么 DB2 就会从磁盘文件中读取相关数据页。

图 3.11 介绍了 Group Buffer Pool 使用的实例。

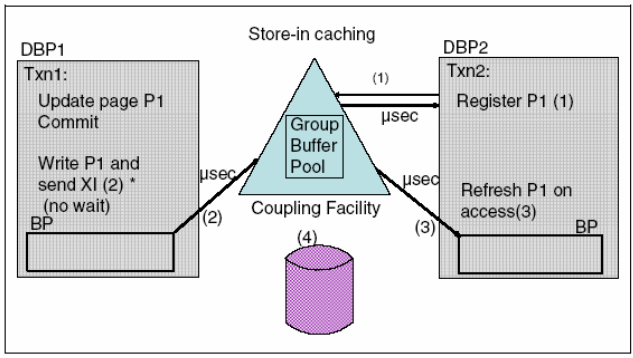


图 3.11　DB2 Group Buffer Pool 使用实例

图 3.11 假设 DB2 MEMBER DBP1 中的交易 Txn1 先执行，并且已经将相关的数据页 P1 读到 Local Buffer Pool 中。

（1）DB2 MEMBER DBP2 中的交易 Txn2 也需要访问数据页 P1，那么 DBP2 需要将对 P1 的访问信息记录到 CF Group Buffer Pool 中的某个 Directory entry 中。

（2）DBP1 中的交易 Tnx1 在更新完 P1 数据，做 Commit 的时候，DB2 会将更新后的 P1 数据写入 CF Group Buffer Pool 中的某个 Data element 中，同时，CF 会由 Group Buffer Pool 中的某个 Directory entry 了解到 DBP2 中的交易也要访问数据页 P1，并通过“Cross-invalidation”过程，将 DBP2 的 Local Buffer Pool 中的数据页 P1 置为“不可用（Invalidate）”状态。

（3）Tnx2 交易需要访问数据页 P1，但是发现此时 Local Buffer Pool 中的数据页 P1 已经是“不可用（Invalidate）”状态，那么 DB2 DBP2 就会从 CF Group Buffer Pool 的 Data element 读取最新的数据页 P1 到 Local Buffer Pool。

（4）经过一段时间，CF Group Buffer Pool Data element 中的数据会通过“Castout”动作，将数据写到磁盘文件中。

3.2.3　DB2 Data Sharing的连续可用性

DB2 Data Sharing 的设计目标就是当某一个硬件或软件不可用的时候，仍然可以保证 DB2 的可用。

在 DB2 Data Sharing 体系中，可以消除以下各种以前的单点故障：某个 DB2 子系统异常、某个 z/OS LPAR 异常、某个物理主机 CPC 异常、某个 IO 通道异常。另外，由于 DB2 Data Sharing 是建立在 Parallel Sysplex 基础上的，而 Parallel Sysplex 本身就包括 2 个时钟控制器（Timer）、2 个 CF、多条 CF 与 LPAR 的连接通道，因此 DB2 Data Sharing 也可以避免由于以上某个硬件引起的故障。

DB2 用户经常会对某个 DB2 MEMBER 或某个 z/OS LPAR 进行系统维护，需要下宕 DB2 MEMBER 或某个 z/OS LPAR。在 DB2 Data Sharing 环境中，这些维护工作将不再对 DB2 的连续可用性造成任何影响。相关实现原理如图 3.12 所示。

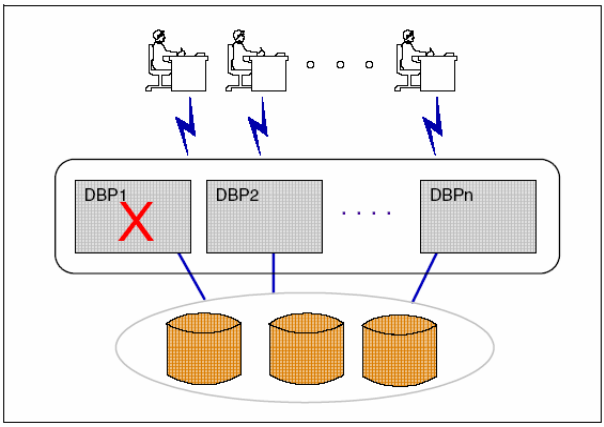


图 3.12　对某个 DB2 MEMBER/z/OS LPAR 维护

当用户需要对 Data Sharing 中某个 DB2 MEMBER 下宕进行维护前，可以将目前该 DB2 MEMBER 中的交易移到其他的 DB2 MEMBER 上，在此 DB2 MEMBER 没有任何交易的情况下，下宕此 DB2 MEMBER，对其进行维护，之后再将该 DB2 MEMBER 启动。由于相关交易可以到其他的 DB2 MEMBER 中执行，因此这个操作对用户没有任何影响。同理，如果需要对某个 z/OS LPAR 进行启下维护，也可以使用上面类似的方法。如果要对 Data Sharing 中的全部 DB2 MEMBER 进行启下维护操作，可以按照上面的方法一个（DB2 MEMBER）一个（DB2 MEMBER）地做，这样对用户使用也没有影响。我们将这个操作称为“循环维护（rolling maintenance）”或“循环启下（rolling IPL）”。

当用户需要对 Parallel Sysplex 中的 CF 进行启下维护操作时候，也可以使用上面所介绍的方法，具体说明如图 3.13 所示。

但对某个 CF 下宕前，需要将其上的 DB2 Data Sharing 的 Structure（包括上面提到的

Lock Structure 和 GBP Structure）转移到另一个 CF 中。对这个 CF 维护操作完毕，重新启动 CF 后，可以将之前移走的 DB2 Data Sharing 的 Structure 移回到此 CF 中。

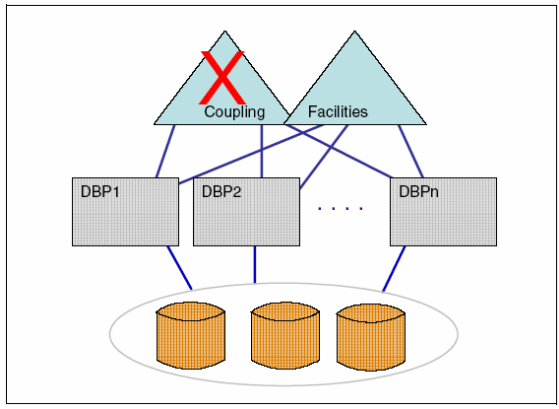


图 3.13　对某个 CF 维护

3.2.4　异常情况对DB2 Data Sharing的可用性影响

前面提到在 DB2 Data Sharing 中，用户对某个 DB2 MEMBER 或某个 z/OS LPAR 进行系统维护，只需要下宕 DB2 MEMBER 或某个 z/OS LPAR，维护完毕后，启动相关 DB2 MEMBER 或 z/OS LPAR 即可，对 DB2 的连续可用性没有任何影响。

当某个 DB2 MEMBER 或某个 z/OS LPAR 异常下宕时，可能会对用户使用造成影响，如图 3.14 所示。

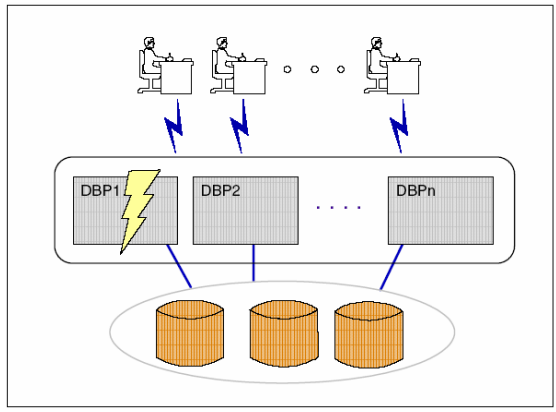


图 3.14　DB2 MEMBER 异常下宕对用户的影响

如图 3.14 所示，当 DB2 MEMBER DBP1 异常下宕后，其他 DB2 MEMBER 中的应用程序还是可以继续执行的，新上来的应用程序，会被分配到此 Data Sharing 中其他的 DB2 MEMBER 中执行。

而在 DB2 MEMBER DBP1 异常下宕时，其中还没有执行完毕的程序可能拿住一些表的锁，而这些锁如果是上面提到的 Data Sharing 中的全局锁（Global Locks），那么 Data Sharing 会在 CF 的 Lock Structure 中将这些锁的状态变为保留锁（Retained Locks），其他所有应用程序将不能再访问被 Retained Locks 保护的表，直到这部分锁被释放为止。为此，用户需要迅速重启此 DB2 MEMBER，来释放这部分锁，保证其他应用程序可以正常访问相关表。

下面两种情况是用户经常会遇到的：

- 当某个 DB2 MEMBER 异常下宕，而其所在的 z/OS LPAR 没有问题的时候。需要迅速手工重启此 DB2 MEMBER，或者使用系统自动重启工具 Automatic Restart Manager（ARM）来重启此 DB2 MEMBER。在 DB2 MEMBER 重启后，将会释放 Retained Locks，保证应用程序对表的正常访问。
- 当某个 z/OS LPAR 异常下宕，其上的 DB2 MEMBER 也会被下宕。为了节省时间，此时需要将此 DB2 MEMBER 迅速启动到其他的 z/OS LPAR，以便尽快释放 Retained Locks 保证应用程序对表的正常访问。当此 z/OS LPAR 重新启动后，再将此 DB2 MEMBER 重新启回此 z/OS LPAR 即可。



3.3 DB2 Data Sharing 的实现

本节介绍的是在 DB2 Data Sharing 实施过程中需要注意的几个问题，主要包括如下方面的内容：

- 命名规则
- DB2 日志
- DSNZRAM 参数设置

3.3.1 命名规则

在搭建一个 DB2 Data Sharing 前，需要仔细考虑相关命名规则。一套良好的命名规则，

可以帮助数据库管理员方便地区分、管理整个 Data Sharing Group 和其中的每个 DB2 MEMBER。

下面介绍如下 Data Sharing 命名：

- Data Sharing Group 命名
- DB2 MEMBER Subsystem（SSID）命名
- Log 文件命名
- Bootstrap 文件（BSDS）命名
- DB2 Catalog 和 Alias 命名
- Temp work file 命名
- CF STRUCTURE 命名

3.3.1.1　Data Sharing Group命名

Data Sharing Group 名是 Parallel Sysplex 中 Data Sharing 的唯一标识。Data Sharing Group 名需要写到 Parallel Sysplex 中的 Coupling Facility Resource Management（CFRM）。在 CF 中的三类 DB2 Structure 命名都要以 Data Sharing Group 名为前缀，目的是将每一个 Data Sharing Group 和与之对应 CF Structure 关联起来。例如，groupname_LOCK1，groupname_SCA，groupname_GBP0 等。

Group 名最多为 8 个字符，可以包括 A～Z，0～9，$，#和@，但必须以 A～Z 字符开头。为了避免与 XCF group 命名冲突，DB2 Data Sharing Group 名不以字母 A～I 开头（“DSN”开头的例外），不以字符串“SYS”开头，不以字符串“UNDESIG”作为 DB2 Data Sharing Group 名。

例如，DSNPROD、DSNPRD1、DSNPRD2 等都可以作为 DB2 Data Sharing Group 名，而 DB2 PRD1 就不能作为 DB2 Data Sharing Group 名。

3.3.1.2　DB2 MEMBER Subsystem（SSID）命名

SSID 即为 Data Sharing Group 的每个 DB2 MEMBER 名称。MEMBER 命名最多为 8 个字符。可以包括 A～Z，0～9，$，#，@，但必须以 A～Z 字符开头。每个 MEMBER 的命名与所在的 group 命名存在一定的联系。

一般而言，SSID 命名规则如下：AAxA，其中 x 用于标识该 DB2 MEMBER 所在的 Data Sharing Group 序列号，用 1～9 或 A～Z 来表示。例如 JB0A 是一个 Data Sharing Group 的

命名，JB1A 是 DSNJB0A 这个组的第一个 MEMBER 名。确保 Data Sharing Group 中每个 DB2 MEMBER 的命名规范及连贯性有利于日常维护和降低管理的复杂性。

3.3.1.3 Log文件命名

Data Sharing Group 中每个 DB2 MEMBER 都有各自的 Log 文件。DB2 Log 文件又分为 Active Log 文件和 Archive Log 文件。为了可以更好地分辨和管理 Log 文件，一般在 Log 文件的命名中，通常要加入 Data Sharing Group 名和 SSID 名作为 Log 文件命名的一部分。例如，某个 Active Log 文件命名“groupname.ssid.LOGCOPY1.DS01”，某个 Archive Log 文件命名“groupname.ssid.ARCHLOG1”。

3.3.1.4 Bootstrap文件（BSDS）命名

Data Sharing Group 中每个 DB2 MEMBER 都有各自的 Bootstrap 文件（BSDS）。BSDS 包含了 Data Sharing Group 内其他 DB2 MEMBER 的相关信息，如其他 DB2 MEMBER 的命名、其他 DB2 MEMBER 使用的 Log 文件和 Archive Log 文件命名等信息。例如，当某个 DB2 MEMBER 需要恢复某张表数据时，就要读取其他 DB2 MEMBER 的 Bootstrap 文件（BSDS）信息。为了防止某个 Bootstrap 文件（BSDS）损坏，需要为每个 DB2 MEMBER 定义一对 Bootstrap 文件（BSDS），其命名规则通常如下：groupname.ssid.BSDS01 和 groupname.ssid.BSDS02。

3.3.1.5 DB2 Catalog和Alias命名

Data Sharing Group 中，需要考虑每个 DB2 MEMBER 的数据库系统文件的命名和共享应用数据的命名。

每个 DB2 MEMBER 的数据库系统文件一般为：DSN groupname.ssid.**。

共享应用数据的命名一般为：DB2 groupname.**。

3.3.1.6 DB2 Temp work file命名

Data Sharing Group 中，每个 DB2 MEMBER 需要有各自的临时文件 Temp Work File，用来在处理数据的时候做排序临时文件。这些文件命名中一般包括“WRK”字段，用以区分是 Temp Work File，例如：groupname.WRKssid.**。

3.3.1.7 CF STRUCTURE命名

Cache Structure（GBP）命名与 DB2 Data Sharing group 的命令有一定联系，其命名规

则通常如下：DSNxxxx_GBPy。其中 xxxx 为 group name，y 为数字。如：DSNPB01_GBP1（代表 PB01 数据库的 GBP1 Structure）。

List Structure（SCA）命名规则通常如下：DSNxxxx_SCA。其中 xxxx 为 group name。如：DSNPB01_SCA（代表 PB01 数据库的 SCA Structure）。

Lock Structure（LOCK）的命名规则通常如下：DSNxxxx_LOCK1。其中 xxxx 为 group name。如：DSNPB01_LOCK1（代表 PB01 数据库的 LOCK Structure）。

3.3.1.8　其他命名

1. Command Prefix 命名

Data Sharing Group 中使用 DB2 Command 的时候，需要在 Command 前加上一段前缀，这就是 Command Prefix。Command Prefix 一般命名为-ssid。

例如，-DBP1。

2. DSNZPARM 命名

Data Sharing Group 中 DB2 系统参数 DSNZPARM 的命名一般以“DSNZ”字符串开头，具体命名规范为：DSNZssid。

3. IRLM Group 命名

Data Sharing 中，需要为整个 Data Sharing Group 在 XCF 中定义一个 IRLM Group 命名，一般以“DXR”开头，具体命名为：DXR groupname。

4. IRLM 命名

Data Sharing Group 中，每个 DB2 MEMBER 需要有一个对应的 IRLM 名，此 IRLM 名一般以“IR”开头。

3.3.1.9　命名规范示例

表 3.1 显示的是 DB2 Data Sharing Group 相关命名规则示例，供用户参考。

表 3.1　命名规范示例

对　象	举　例
z/OS 系统名	SYSA, SYSB
DB2 共享组名	DSNDB2P
DB2 共享组连接名	DB2P（使用用户定义的 DB2 SSID）
DB2 子系统名	DBP1, DBP2
命令前缀	–DBP1, –DBP2

续表

对　象	举　例
IRLM 子系统名	IRP1,IRP2
IRLM XES 组名	DXPDB2P
系统编目和别名	DSNDB2P，用户可以自行定义，需要提示的是如果用户使用 BACKUP SYSTEM 和 RESTORE SYSTEM DB2，需要对 LOG、BSDS 和其他 Database 分别定义不同的独立的 ICF 系统编目
系统装载参数模块名	DSNZDBP1, DSNZDBP2
临时工作文件	WRKDBP1, WRKDBP2
DB2 Location 名	DB2PLOC（使用现有的 DB2 SSID Iocation）
DB2 通用 LU 名	LUDB2P（使用现有的 DB2SSID LU 名称）
DB2 网络 LU 名	LUDBP1, LUDBP2
BSDS 名	DSNDB2P.DBP1.BSDS01 DSNDB2P.DBP1.BSDS02
日志文件	DSNDB2P.DBP1.LOGCPY1.DS01 DSNDB2P.DBP1.ARCLG1.Dyyddd.Thhmmsst.Axxxxxxx

3.3.2　DB2 日志

DB2 日志是数据库记录自身运行情况的相关文件，在 Data Sharing Group 中每个 DB2 MEMBER 都有各自的 Active Log 和 Archive Log。

3.3.2.1　Active Log

在 Data Sharing Group 中每个 DB2 MEMBER 都有各自的 Active Log，DB2 的 Active Log 都是成对出现的，每对 Active Log 由 Logcopy1 和 Logcopy2 两个文件组成，其内容是完全一致的。在 Data Sharing 中，推荐 Logcopy1 和 Logcopy2 两个 Active Log 文件放在不同磁盘上，避免由于某个磁盘发生错误，而造成 Active Log 文件无法访问的情况。Data Sharing Group 中每个 DB2 MEMBER 的 BSDS 文件也需要进行类似的考虑。

3.3.2.2　Archive Log

在某个 DB2 MEMBER 的 Active Log 文件写满的时候，DB2 会自动将此 Active Log 文件中的内容复制到相应的 Archive Log 文件中。在磁盘空间充裕的情况下，建议将 Archive Log 也保存到磁盘上，因为 DB2 恢复动作往往也需要访问此 Archive Log 文件。当所有 DB2 Log 都在磁盘上时，恢复速度会更快。

3.3.3　DB2 Data Sharing重要参数

下面将介绍一些 DB2 系统参数 ZPARM，DB2 IRLM 参数。

3.3.3.1　一些 Data Sharing重要系统参数

- DSHARE（YES/NO）：表示此 DB2 MEMBER 是否是 Data Sharing Group，YES 表示此 DB2 MEMBER 存在于一个 Data Sharing Group 中。
- GRPNAME：表示 Data Sharing Group 名。
- MEMBNAME：表示 DB2 MEMBER 名。
- SSID：表示 DB2 MEMBER 的 SSID 名。

3.3.3.2　一些重要的DSNPARMS参数

- CTHREAD：表示能够连接到某个 DB2 MEMBER 的最大线程数（包括 CICS、BATCH 和 TSO 等所有连接到 DB2 的线程最大数），此参数是影响 DBM1 地址空间虚存消耗的关键数值之一。
- LOGAPSTG：在 Data Sharing Group 某个 DB2 MEMBER 中进行恢复动作的时候可以使用的 DBM1 中内存的大小，其默认数值为 100MB，该数值影响 DB2 MEMBER 的恢复速度，建议使用 100MB。
- SYNCVAL：表示 DB2 MEMBER 的 statistics 报表收集的间隔时间，默认值为 5 分钟，建议使用 1 分钟。
- PCLOSEN：表示 DB2 中某个数据文件在多少个 Checkpoint 后，如果一直没有被更新过，那么 DB2 就会将其由 RW 状态改为 RO 状态，默认值是 9 个 Checkpoint 后会自动改为 RO 状态。
- PCLOSET：表示 DB2 中某个数据文件在多长时间后，如果一直没有被更新过，那么 DB2 就会将其由 RW 状态改为 RO 状态，默认值是 10 分钟后会自动改为 RO 状态。
- CHKFREQ：表示 DB2 在多长时间或写入多少条 Log 数据后，会做一次 Checkpoint。建议使用 5 分钟。
- URCHKTH：表示 DB2 中如果某个程序在多少个 Checkpoint 后，如果还没有做过 Commit，就会在 Console 中写一条警告信息，默认值为 0。

3.3.3.3 一些重要的IRLM参数

- IRLMGRP：表示 IRLM Group 名；
- IRLMNM：表示某个 DB2 MEMBER IRLM 名；
- IRLMID：表示此 DB2 MEMBER 在 IRLM Group 中的序号，由 1 开始；
- SCOPE：其值分为 GLOBAL 和 ONLY，在 DB2 Data Sharing 中，需要指定 GLOBAL 参数。



3.4 DB2 Data Sharing 的恢复

本节主要介绍以下内容：

- DB2 Data Sharing 中某个 DB2 表的恢复；
- DB2 Data Sharing 中某些组件异常的恢复；
- SFM 和 ARM 功能的介绍；
- DB2 特殊启动模式“Light”的介绍。

3.4.1 对某个DB2 表的恢复

对于 DB2 Data Sharing 中某个对象的恢复，需要使用 DB2 RECOVER Utility，其恢复的原理是先找到此表的最后一次备份（Image Copy），将此备份恢复到该表，之后扫描数据库的日志（Log），找到在此表最后一次备份之后所执行的所有更新和删除操作，按照操作的时间顺序逐一地追加到该表中。与 Stand Alone 的 DB2 所不同的是，在 DB2 Data Sharing 进行日志扫描的时候，需要扫描此 Data Sharing Group 中所有 DB2 MEMBER 的日志信息，并将这些信息按照时间先后顺序排列。

下面将介绍一些术语：

- Log Record Sequence Number（LRSN）
- Group Buffer Pool Recovery Pending（GRECP）
- Logical Page List（LPL）

3.4.1.1 Log Record Sequence Number（LRSN）含义

与以前的 Stand Alone 的 DB2 日志信息（RBA）相比，在 DB2 Data Sharing 中，每一

条日志信息，都会增加相应的 DB2 MEMBER 名，结合从 Sysplex Timer 中得到的时间信息（Timestap），形成一个完整的 Data Sharing 日志信息（LRSN）。因此，在 DB2 Data Sharing 中，每条 LRSN 信息都是唯一的。

3.4.1.2　Group Buffer Pool Recovery Pending（GRECP）含义

在 DB2 Data Sharing 中，如果 CF 发生异常，或者如果每个 DB2 MEMBER 失去了与 CF GBP 的连接，那么 DB2 会自动将在 CF GBP 中保存的相关数据页（Page）置为 Group Buffer Pool Recovery Pending（GRECP）状态。要恢复 GRECP 状态的数据页，DB2 需要扫描并追加数据库日志中记录的与此数据页相关的所有记录。

3.4.1.3　Logical Page List（LPL）含义

在 DB2 Data Sharing 中，如果无法从 CF GBP 中读取或向 CF GBP 中写入相关数据页，DB2 会将此数据页置为 LPL 状态。以前的 Stand Alone 的 DB2 中，如果 DB2 无法从磁盘上读取或者无法将数据写到磁盘上，也会将数据页置为 LPL 状态。在 DB2 中，“DSNB250E”、“DSNB303E”、“DSNB311I”和“DSNB312I”等信息都表明 DB2 将某个数据页置为 LPL 状态。

3.4.1.4　对GRECP/LPL状态的恢复方法

一般用户只会在下面三种情况下遇到 GRECP/LPL 状态的数据：

- 灾难恢复测试演练的时候；
- 整个 Parallel Sysplex 系统全部不可用的情况下；
- 磁盘连接或 CF 连接不可用的情况。

在系统恢复后，重启 Data Sharing 的过程中，DB2 会自动将部分数据页置为 GRECP/LPL 状态。之后就需要用户手工解除此状态，具体方法可以有下面两个：

- 使用 DB2 Recover Utility，此方法比较复杂，不建议使用；
- 使用 DB2“START DATABAS”命令，此方法比使用 Recover Utility 恢复方法要简便很多，推荐使用。在发出“START DATABASE”命令后，DB2 就会从日志中扫描并追加相关信息，来解除此状态。

为了加快 DB2 日志扫描的速度，从而加快对于 GRECP/LPL 状态表的恢复速度，建议将 DSNZPARM 中的参数 LOGAPSTG 值置为最大值 100MB，此参数的含义是在进行日志扫描的时候，可以使用的内存的大小。

3.4.1.5 对GRECP/LPL状态的恢复几点建议

- 建议将 DSNZPARM 的 LOGAPSTG 设置为 100MB；
- 确保数据库日志（Log）文件的可用性；
- 尽量使用 DB2“START DATABASE”命令方式去解除 GRECP/LPL 状态的表。

3.4.2 对某些组件异常的恢复

使用 Parallel Sysplex 和 DB2 Data Sharing 的最大好处是提高系统和数据库的可用性。用户可以通过使用、调整系统 CFRM（Coupling Facility Resource Management）、SFM（Sysplex Failure Management）和 ARM（Automatic Restart Management）的策略来减少某个系统或某个 DB2 MEMBER 异常对整个 Data Sharing 的影响。

下面介绍发生以下几种异常情况下的恢复流程：

- 某个 DB2 MEMBER 发生异常；
- 某个 z/OS LPAR 发生异常；
- 某个 CF 发生异常。

3.4.2.1 对某个DB2 MEMBER异常的恢复

当 DB2 Data Sharing 中的某个 DB2 MEMBER 发生异常的时候，此 DB2 MEMBER 上还没有释放的锁（Lock）会转变成保留锁（Retained Lock），被保留锁（Retained Lock）锁住的资源，Data Sharing 中其他 DB2 MEMBER 也不能访问。因此，如果发生这种情况，需要用户尽快重启此 DB2 MEMBER，来释放保留锁（Retained Lock）。

当某个 DB2 MEMBER 发生异常时，建议用户使用 Automatic Restart Manager（ARM）来实现 DB2 MEMBER 的自动重启，这样比手工启动 DB2 MEMBER 的响应速度快很多，来达到尽快释放保留锁（Retained Lock）的目的。

3.4.2.2 对某个ZOS LPAR异常的恢复

当 Parallel Sysplex 中的某个 LPAR 发生异常，此 LPAR 上运行的 DB2 MEMBER 也会异常下宕，此时就要通过 Automatic Restart Manager（ARM）来自动将此 DB2 MEMBER 启动到其他正常的 LPAR 上，以便尽快释放保留锁（Retained Lock）。当此 LPAR 恢复后，再将 DB2 MEMBER 启回此 LPAR 即可。

3.4.2.3　对CF的恢复

建议用户在 Parallel Sysplex 中使用两个 CF，将 DB2 Data Sharing 的 CF Structure 使用 DUPLEX 方式定义到两个 CF 中，这样如果一个 CF 出现问题，那么其上的所有 Structure 会 Rebuild 到另一个 CF 中，从而保证了 DB2 Data Sharing 的可用性。

当两个 CF 都出现故障时，CF 中的 SCA 和 LOCK1 都不可用，这使得 Data Sharing 中的所有 DB2 MEMBER 都会异常下宕。当 CF 的 Structure 恢复后，就要重启整个 Data Sharing 中的所有 DB2 MEMBER 以完成整体重启（Group Restart）以重建 CF 中的 SCA Structure 和 LOCK1 Structure。

3.4.3　Sysplex Failure Management（SFM）

Sysplex Failure Management（SFM）的功能是通过预先设定一些策略，从而定期监控整个 Parallel Sysplex 中各个 ZOS LPAR 的运行情况，如果某个 LPAR 出现异常，SFM 就会根据预先设定的策略，自动采取一些动作，将此 LPAR 从整个 Parallel Sysplex 中隔离，避免影响其他的 LPAR，保证了整个 Parallel Sysplex 的可用性，进而保证了其上运行的 DB2 Data Sharing 的可用性，建议用户使用此功能。

3.4.4　Automatic Restart Management（ARM）

Automatic Restart Management（ARM）的功能是通过预先设定的策略，实现监测到某个 DB2 MEMBER 异常下宕的情况下，可以立即自动重启此 DB2 MEMBER，以便尽快释放此 DB2 MEMBER 中的保留锁（Retained Lock），提高整个 DB2 Data Sharing 的可用性。建议用户使用此功能。

3.4.5　DB2 MEMBER Light 模式重启

当 Parallel Sysplex 中某个 ZOS LPAR 出现异常的情况下，此 LPAR 上的 DB2 MEMBER 也会异常下宕，为了尽快释放此 DB2 MEMBER 中的保留锁（Retained Lock），我们一般会将此 DB2 MEMBER 启动到其他正常的 LPAR 上，但是此 LPAR 可能由于容量限制，不能保证此 DB2 MEMBER 所需资源，我们在此 DB2 MEMBER 启动成功释放保留锁（Retained Lock）后，还需要手工下宕此 DB2 MEMBER。

为此，在 DB2 Data Sharing 中，提供了一种特殊的 DB2 MEMBER 的启动模式“Light”。使用此模式启动 DB2 后，此 DB2 MEMBER 会在启动释放保留锁（Retained Lock）后，自动下宕。



3.5 课后习题

1．简述 DB2 Data Sharing 技术的主要优点是什么。
2．简述 Parallel Sysplex 系统的主要由哪几个组件构成。
3．简述 DB2 Data Sharing 是如何控制并发性和保证数据一致性的。
4．简述 DB2 Data Sharing 中 Active Log 与 Archive Log 的区别。
5．简述 DB2 Light 启动模式的功能。

第 4 章　DB2 常用Utility基础

本章为 DB2 常用 Utility 基础，主要介绍 DB2 产品自带 Utility 方面的知识，内容涉及 Utility 的调用方法、Online Utility 和 Offline Utility 的介绍等。



4.1　DB2 Utility 简介

4.1.1　什么是Utility

数据库 DB2 家族产品中，这些 Utility 作为可选择的产品，可以供客户单独购买使用。这些 Utility 可以理解为作用在 DB2 子系统上或 DB2 OBJECT 上，是维护与管理的 DB2 数据库子系统的必要工具。

4.1.2　Utility的分类

DB2 Utility 可分为两种类型：Online Utility 和 Stand Alone Utility。DB2 Online Utility 的运行方式类似于 MVS 的批量作业或一个存储过程，要求 DB2 处于运行状态。DB2 Stand Alone Utility 并不要求 DB2 处于运行状态，但这些 Utility 只能以 MVS JCL 的方式运行。

4.1.3 调用Utility的方法

调用 DB2 Utility 有以下四种方式：

- 在 IBM DB2I 工具中使用 DB2 Utility PANEL 生成 JOB
- 在 TSO 环境下使用 DSNU CLIST 命令去调用 DB2 Utility
- 使用 JCL 去调用 Utility DSNUPROC PROCEDURE
- 使用手工编辑 JCL 去执行 DSNUTLIB

一般我们常用的是第三和第四种方式，在日常的工作中掌握使用这些 DB2 Utility 对于使用、维护、管理 DB2 数据库子系统环境有非常重要的意义，也是必不可缺的。

4.2 DB2 Online Utility

4.2.1 如何调用DB2 Online Utility

4.2.1.1 控制语句书写规则

Online Utility 的控制语句一般从 SYSIN 文件集中读入，LISTDEF 的控制语句从 SYSLISTD 中读入，TEMPLATE 控制语句从 SYSTEMPL 中读取，这些控制语句必须遵循下面的原则：

- 如果控制语句输入文件是定长（FB）80 格式的，则自第 73 列之后的语句将被忽略；
- 因为控制语句在解析时会要进行连接，所以控制语句中的语句可以换行而不需要连接符；
- 在一个文件集中的控制语句必须使用同一个字符集。
- 一个 SYSIN 控制语句中可以包含多个 Utility 语句。

4.2.1.2 DB2 Online Utility所需的DATA SET

每个 Online Utility JOB 都需要一个 SYSIN DD 语句，用于描述输入 DATA SET，而一些 Utility 同时还需要使用其他一些 DATA SET，作为参数的输入。具体内容请参考《DB2 UDB for z/OS V8 Utility Guide and Reference》。

4.2.2　BACKUP SYSTEM Utility

4.2.2.1　Utility功能简介

BACKUP SYSTEM Utility 通过调用 z/OS　DFSMShsm（V1.5 以上版本）对 DB2 数据及日志所在的物理卷进行拷贝以实现对整个系统的数据备份，包括 DB2 数据及 LOG 信息。注意，使用该 Utility 进行系统备份，要求所有需要进行备份的 DATA SET 都是 SMS 管理的 DATA SET。之后可利用 Restore Utility 对这些备份数据进行恢复。

BACKUP SYSTEM Utility 使用拷贝池（COPY POOL）对数据卷进行备份。每个 DB2 子系统都会拥有两个以上的 COPY POOL，一个用于数据库备份，一个用于 LOG 的备份。

【输出】DB2 数据和 LOG 信息所在卷的备份，该 Utility 的历史信息记录在 BSDS 中。

【运行的阶段】如表 4.1 所示。

表 4.1　运行的阶段

阶　段	描　述
UTILINIT	初始化 Utility
COPY	备份数据
UTILTERM	清除

4.2.2.2　控制语句及语法

【语法】请参考《DB2 UDB for z/OS V8 Utility Guide and Reference》。

【主要控制语句描述】

- **FULL** 对数据库 COPY POOL 和 LOG COPY POOL 进行备份。默认值是 FULL。

在此之前需确认数据库 COPY POOL 中已经包含了数据库文件所在的卷及与数据库相关的 ICF Catalog 所在的卷；LOG COPY POOL 包含了 BSDS、Active LOG 及相关 Catalog 所在的卷。

- **DATA ONLY** 只对数据库 COPY POOL 进行备份。

同样，需要确认数据库 COPY POOL 中已经包含了数据库文件所在的卷，以及与数据库相关的 ICF Catalog 所在的卷。

4.2.2.3　如何运行BACKUP SYSTEM Utility

作业中使用的 DATA SET 如表 4.2 所示。

表 4.2　作业中使用的 DATA SET

DATA SET	描　　述	是否必需
SYSIN	包含 Utility 控制语句的文件	是
SYSPRINT	输出 MESSAGE 的文件	是

4.2.2.4　参考作业流

1. 对一个 DB2 子系统或一个 Data Sharing Group 做全备份

```
//STEP1   EXEC DSNUPROC,TIME=1440,
//        UTPROC='',
//        SYSTEM='DSN'
//SYSIN   DD *
   BACKUP  SYSTEM
/*
```

2. 对一个 DB2 子系统或一个 Data Sharing Group 做 DATA ONLY 备份

```
//STEP1   EXEC DSNUPROC,TIME=1440,
//        UTPROC='',
//        SYSTEM='DSN'
//SYSIN   DD *
   BACKUP  SYSTEM  DATA ONLY
/*
```

4.2.3　COPY Utility

4.2.3.1　Utility功能简介

COPY Utility 可以对下面任何一个 OBJECT 做 IMAGE COPY：

- TABLESPACE
- TABLESPACE PARTITION
- DATA SET OF LINEAR TABLESPACE
- INDEX SPACE
- INDEX SPACE PARTITION

IMAGE COPY 有如下两种类型。

- FULL IMAGE COPY：对 TABLESPACE，PARTITON，INDEX SPACE 的所有 PAGE。
- INCREMETAL IMAGE COPY：只对最后一次 FULL IMAGE COPY 以来修改的

PAGES 做 COPY。

这些备份通常使用 RECOVERY UTILIITY 进行恢复，也可以用来做 UNLOAD、MERGECOPY，COPYTOCOPY 等操作。

【输出】

- 最多只能允许四个顺序文件用于存放 IMAGE COPY 的内容；
- 在 SYSIBM.SYSCOPY 表中插入描述 IMAGE COPY DATA SET 的记录；
- 如果指定了 CHANGELIMIT 选项，将输出 TABLESPACE 变化情况的报告。

【运行阶段】如表 4.3 所示。

表 4.3　运行阶段

阶　段	描　述
UTILINIT	初始化 Utility
REPORT	CHANGELIMIT 的 REPORT
COPY	对 OBJECT 做 COPY
UTILTERM	清除相关内容

4.2.3.2　控制语句和语法

【语法】请参考《DB2 UDB for z/OS V8 Utility Guide and Reference》。

【主要控制语句描述】

- **TABLESPACE：**指定需要做 COPY 的 TABLESPACE 的名字。
- **INDEXSPACE：**指定需要做 COPY 的 INDEXSPACE 的名字。
- **INDEX：**指定需要做 COPY 的 INDEX 的名字
- **DSNUM：**对于 TABLESPACE，指定 TABLESPACE 的第几个 PARTITION 或 DATA SET 需要做 COPY，否则对整个 TABLESPACE 做 COPY；对于 INDEXSPACE，指定 INDEXSPACE 的第几个 PARTITION 做 COPY，否则 COPY 整个 INDEXSPACE。
- **COPYDDN ddname1,ddname2：**指定存放 IMAGE COPY 的主要（DDNAME1）和备份（DDNAME2）文件的名字。也可以是一个 TEMPLATE 的名字。
- **PARALLEL (num-object)：**指定 COPY 的并行度，即同时可以有几个 OBJECT 做 COPY，线程数为 2*N+1，N 为 OBJECT 的数量。如果没有指定 PARALLEL 参数，DB2 对于每个单独的 OBJECT 使用三个线程。
- **TAPEUNITS(num-tape-units)：**指定并行 COPY 中 Utility 可以动态使用的 TAPE

DRIVERS 的数量。

- **FULL**：决定使用 FULL COPY（全量备份）还是 INCREMENTAL COPY（增量备份）。

 YES：使用 FULL COPY，默认值。

 NO：使用 INCREMENTAL COPY，只备份最近一次 FULL COPY 以来的修改增量部分，对于 COPY INDEX 无效。

 在下面几种情况下，不可以使用 INCREMENTAL COPY：

 （1）最后一次 FULL COPY 时使用了 CONCURRENT 选项；

 （2）TABLESPACE 没有做过 FULL COPY；

 （3）TABLESPACE 做了 LOAD 或 REORG，而且其中没有做 INLINE 的 COPY；

 （4）不可以对 DSNDB01.DBD01， DSNDB01.SYSUTILX，DSNDB06.SYSCOPY 做 INCREMENTAL COPY。

- **SHRLEVEL**：指定 COPY 期间，其他的程序是不是可以访问该 TABLESPACE 或 INDEX。

 REFERENCE：只允许只读的访问。默认值。

 CHANGE：允许其他程序对该 TABLESPACE 或 INDEX 进行修改。当你使用了 CHANGE OPTION 时，没有经过 COMMIT 的记录也会被 COPY 下来，对连续运行要求高的系统环境建议采用此方式以降低对外服务的影响。

4.2.3.3 如何运行COPY Utility

1. 运行前的准备工作

- 当 TABLESPACE 处于 CHECK-PENDING 状态时，不能对它做 IMAGECOPY；
- 当 TABLESPACE 处于 COPY-PENDING 状态时，需要对它做 FULL IMAGE COPY。

2. 作业中的重要 DATA SET

如表 4.4 所示。

表 4.4 作业中重要的 DATA SET

DATA SET	描　　述	是否必需
SYSIN	包含 COPY Utility 控制语句的文件	是
SYSPRINT	Utility 的输出信息	是
COPIES	1～4 个包含 IMAGE COPY 的输出文件。在作业的控制语句中，它们的 DD NAME 是用 COPY DDN 和 RECOVERDDN 指定的	是

3. CATALOGING IMAGE COPY 文件

使用 DISP=(NEW,CATLG,CATLG)对以 COPYDDN 命名的 IMAGE COPY 文件进行 CATALOG。如果文件是 CATALOGED，IMAGE COPY 完成后，在 SYSIBM.SYSCOPY 中插入一行记录，而该记录中的 DSVOLSER 字段是空白的。因此如果另一个备份 Utility 备份时使用了同样的文件名，即使该备份文件已被删除，Utility 仍将被中止。

4.2.3.4　参考作业流

1. FULL IMAGE COPY

可以对 TABLESPACE，TABLESPACE PARTITION， INDEX SPACE，INDEX SPACE PARTITION 做 FULL IMAGE COPY。

FULL IMAGE COPY 的控制语句如下：

```
COPY TABLESPACE TESTDB.TESTTS
```

建议在下列情况下做一个 FULL IMAGE COPY：

- 在 CREATE 一个新的 OBJECT 后。
- 对已经存在的 OBJECT 做 REORG 或 LOAD RESUME YES 后。

2. INCREMENTAL IMAGE COPY

在下列的条件下可以做 INCREMAENTAL 的 IMAGE COPY：

- 该 TABLESPACE 存在有 FULL 的 IMAGE COPY。
- TABLESPACE 上没有 COPY-PENDING 状态。
- 最近的一次 FULL IMAGE COPY 没有使用 CONCURRENT 选项。
- 如果在某个 PARTITION 上有 FULL IMAGE COPY，我们可以对这个 PARTITION 单独的做 INCREMENTAL IMAGE COPY。

INCREMENTAL IMAGE COPY 的控制语句如下：

```
COPY TABLESPACE TESTDB.TESTTS
 FULL NO
 SHRLEVEL CHANGE
```

TABLESPACE 的 SPACE MAP 记录每个 PAGE 在最后一次 FULL IMAGE COPY 以来的变化情况，因此 INCREMENTAL IMAGE COPY 的速度要比 FULL IMAGE COPY 快得多。

3. 对同一个 TABLESPACE 的不同 PARTITION 分别做 COPY

可以在不同的作业中分别同时备份一个 PARTITION TABLESPACE 的不同 PARTITION，

这样可以大大提升速度，控制语句如下：

```
COPY TABLESPACE NASEDB.NSINSU1  DSNUM 1
```

4. 对一个 OBJECT LIST 做 COPY

可以利用 LISTDEF 和 TEMPLATE 对一组 OBJECT 做 COPY：

```
LISTDEF CPY1 INCLUDE TABLESPACE TESTDB.*
TEMPLATE TMPT UNIT VTAPE RETPD 2
       DSN (BACKUP.PB01.&DB..&TS..P&PART..D&DT.)
       DISP (NEW,CATLG,DELETE) STACK YES
TEMPLATE TMPD
       DSN 'AYX.PB01.&DB..&TS..P&PART.(+1)'
       UNIT 3390
       DISP (NEW,CATLG,DELETE) GDGLIMIT(1)
       MAXPRIME 1000
COPY LIST CPY1 COPYDDN (TMPT,TMPD) PARALLEL (5) TAPEUNITS(5)
SHRLEVEL CHANGE
```

4.2.3.5 性能方面的考虑

到底是选择 FULL IMAGE COPY 还是 INCREMENTAL IMAGE COPY，和 TABLESPACE 中修改的总行数没有关系，而是和有改动的 PAGE 的百分比有关系。如果超过 50%的 PAGES 有改动，则使用 FULL IMAGE COPY 比较合适。在我们日常的工作中，通常采用的是 FULL IMAGE COPY 方式。

4.2.4 LISTDEF Utility

4.2.4.1 Utility功能简介

LISTDEF Utility 可以将多个对象定义到一个 LIST 中。通过 LISTDEF，可以定义一组目标的集合，并且可为该集合定义一个名称。

一条完整的 LISTDEF 控制语句必须包含下列语句：

- LIST 的名称；
- INCLUDE 语句，后面可以跟 INCLUDE 或 EXCLUDE 子句；
- LIST 所包含的 OBJECT 的类型，TABLESPACE 或 INDEXSPACE。

在定义 LIST 时可以使用通配符*来代替所有的 OBJECT。例如，DATABASE-NAME *，TABLESAPSCE *.*等。需要注意的是使用 DEFINE NO 选项定义的 OBJECT 无法使用

LISTDEF 来选择。

【输出】LISTDEF 所包含的 OBJECT 的列表。

【运行阶段】LISTDEF Utility 运行阶段只有 UTILINIT 一个阶段。

4.2.4.2　控制语句及语法

【语法】请参考《DB2 UDB for z/OS V8 Utility Guide and Reference》。

【主要控制语句描述】

- **LISTDEF list-name：**为 DB2 OBJECT 的列表指定一个名称。List-name 最长为 18 个字符（包括字母和数字）。使用时可以在另外的文件中定义该 LIST 也可以在其他的 Utility 语句前定义。
- **INCLUDE：**指定 LIST 中包含的 OBJECT。在该 INCLUDE 语句后面还可以接 INCLUDE 和 EXCLUDE 用来增删 LIST 中的 OBJECT，并支持通配符。
- **EXCLUDE：**在初始的 INCLUDE 语句后，可以用 EXCLUDE 语句删除 LIST 中的某些 OBJECT。
- **TABLESPACES：**生成 TABLESPACE 的列表。
- **INDEXSPACES：**生成 INDEXSPACE 的列表。
- **PARTLEVEL**：指定 LIST 中包含的 TABLESPACE 和 INDEXSPACE 的粒度。对于非 PARTITION 的 TABLESPACE 和 INDEXSPACE，这个参数被忽略。

4.2.4.3　如何使用LISTDEF

1. 和其他 Utility 一起使用

通常我们在一个 Utility 的控制语句之前用 LISTDEF 定义一个 LIST，在后面的 UTITLITY 的控制语句中使用该 LIST。在上一节的 COPY 例子作业中就用到了 LISTDEF：

```
LISTDEF CPY1 INCLUDE TABLESPACE TESTDB.*
```

可以对 TESTDB 下所有的 OBJECT 做 COPY。

2. 使用通配符

在定义 LIST 时可以使用这些通配符（%，*，_，?）。当 DB2 UTILTIY 执行时，会访问 DB2 CATALOG，并动态地展开、解析这些包含通配符的语句。

3. 使用上的限制

不能对和 SYSUTILX 相关的 OBJECT 使用 LISTDEF。不能对 DSNDB06 和 DSNDB01

下的 OBJECT 使用通配符，这些 CATALOG 和 DIRECTORY 下的 OBJECT 必须明确地在 LIST 中写出它们的名字。另外，下面这些 OBJECT 不能被包括在 LISTDEF 定义的 LIST 中：

- TABLESPACE DSNDB01.SYSUTILX
- TABLE SYSIBM.SYSUTILX
- TABLE SYSIBM.SYSUTIL
- INDEXSPACE DSNDB01.DSNLUX01
- INDEXSPACE DSNDB01.DSNLUX02
- INDEX SYSIBM.DSNLUX01
- INDEX SYSIBM.DSNLUX02

4.2.4.4 参考作业流

用文件存放 LIST，并在 QUIESCE Utility 中使用：

在 STEP1 中定义 PS 文件 SHRLIB.DB2.TESTDB.LIST1 存放 LIST NAME1；

在 STEP2 中定义 LIST NAME2，删除 NAME1 中的 TESTTS9 和 TESTTS10。

```
//QUIESCE  JOB 'USER=NAME',
//        TIME=1440,
//        NOTIFY=&SYSUID,
//        CLASS=D,
//        MSGCLASS=X,
//        MSGLEVEL=(1,1)
//*-------------------------------------------------------
//* CREATE AN INPUT DATA SET.
//*-------------------------------------------------------
//STEP1   EXEC PGM=IEBGENER
//SYSPRINT DD DUMMY
//SYSIN    DD DUMMY
//SYSUT2   DD DSN=SHRLIB.DB2.TESTDB.LIST1,
//        DISP=(NEW,CATLG,CATLG),
//        UNIT=3390,SPACE=(4000,(20,20),,,ROUND),
//        DCB=(RECFM=FB,LRECL=80,BLKSIZE=2400)
//SYSUT1   DD *
   LISTDEF NAME1 INCLUDE TABLESPACE TESTDB.*
/*
//*******************************************************
```

```
//* QUIESCE LISTDEF DD LILSTDEF DATASETS
//*****************************************************
//STEP2    EXEC DSNUPROC,UID='Utility.QUIESC2',
//           UTPROC='',SYSTEM='DB1E'
//LISTDSN  DD DSN=SHRLIB.DB2.TESTDB.LIST1,DISP=SHR
//SORTOUT  DD DSN=SHRLIB.DB2.STEP1.SORTOUT,
//         DISP=(MOD,DELETE,CATLG),UNIT=SYSDA,
//         SPACE=(4000,(20,20),,,ROUND)
//SYSIN    DD *
   OPTIONS LISTDEFDD LISTDSN
   LISTDEF NAME2 INCLUDE LIST NAME1
               EXCLUDE TABLESPACE TESTDB.TESTTS9
               EXCLUDE TABLESPACE TESTDB.TESTTS10
   QUIESCE LIST NAME2
/*
```

在输出结果中可以看到,除了 TESTTS9 和 TESTTS10，TESTDB 下的其他 TABLESPACE 都做了 QUIESCE：

```
-PB11 DSNUQUIA – QUIESCE SUCCESSFUL FOR TABLESPACE TESTDB.TESTTS1
-PB11 DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE TESTDB.TESTTS2
-PB11 DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE TESTDB.TESTTS3
-PB11 DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE TESTDB.TESTTS4
-PB11 DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE TESTDB.TESTTS5
-PB11 DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE TESTDB.TESTTS6
-PB11 DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE TESTDB.TESTTS7
-PB11 DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE TESTDB.TESTTS8
-PB11 DSNUQUIA – QUIESCE AT RBA 0002CAFB87EC AND AT LRSN BB0FA564A6E9
```

4.2.5　LOAD Utility

4.2.5.1　Utility功能简介

LOAD Utility 用来将数据 LOAD 到 TABLESPACE 中并 REBUILD INDEX。如果 TABLESPACE 中已有数据，可以选择覆盖或追加。

【运行阶段】如表 4.5 所示。

表 4.5　运行阶段

阶　段	描　述
UTILINIT	初始化 Utility

续表

阶　段	描　述
RELOAD	LOAD 数据并将 INDEX 和 FOREIGN KEYS 写到临时文件。检查每一行上的 CHECK CONSTRINT。在 LOAD 时会做内部的 COMMIT，这样在 Utility 中断重新启动后可以提供 COMMIT 点。 如果在控制语句中指定了 COPYDDN 和 RECOVERYDDN，将创建 INLINE COPIES。 如果控制语句中指定了 SORTKEYS，那么在 RELOAD 阶段开始时会触发一个子任务来控制 INDEX KEYS 的排序。 注意并行的 LOAD 多个 PARTITION 的数据会启动多个 SUBTASK
SORT	如果存在 INDEX 或 FOREIGN KEY，在创建 INDEX 和关联检查前对临时文件中的记录进行排序。 当 RELOAD 阶段发生下列情况时，此阶段会被跳过： • 每个表的 KEY（INDEX KEY 或 FOREIGN KEY）都不超过一个； • 所有的 KEY 都是同一类型的字段(包括 INDEX KEY,INDEX FOREIGN KEY 和非 INDEX FOREIGN KEY)； • LOAD 的数据按照 TABLE 排过序，并且每条记录都 LOAD 到同一个表中。 如果使用了 SORTKEY 参数，SORT 阶段将排过序的 KEYS 存放到内存中在 BUILD 阶段使用来 BUILD INDEX
BUILD	用临时文件中的记录 BUILD INDEX。检查重复键值。对 INDEX 进行 PREFORMAT
SORTBLD	如果在控制语句中指定了进行并行的 BUILD INDEX，所有的 SORT 和 BUILD 都在这个阶段执行
INDEXVAL	纠正重复键值的错误。错误信息来自 SYSERR
ENFORCE	检查表之间的关联关系，并纠正不完整性。错误信息来自 SYSERR
DISCARD	将引起错误的记录 COPY 到 DISCARD 文件
REPORT	如果指定了 ENFORCE CONSTRAINT 或者执行了 INDEX 有效性检查，生成 SUMMARY REPORT，输出到 SYSPRINT 中
UTILTERM	清除 Utility

4.2.5.2　控制语句及语法

【语法】请参考《DB2 UDB for z/OS V8 Utility Guide and Reference》。

【主要控制语句描述】

- **INDDN ddname：**指定输入的数据文件名。文件格式必须是 FIX 或 VARIABLE 的。默认为 SYSREC。
- **RESUME：**指定数据被 LOAD 到空的 TABLESPACE 还是有数据的 TABLESPACE 中。对于非 SEGMENT 的 TABLESPACE，被标记为删除的行或删除的表所占用的空间不会被重用。注意，如果使用了 LOAD RESUME（而不是 PART integer RESUME）会制止不同 PARTITION 上的并行操作。

NO：将数据 LOAD 到一个空的 TABLESPACE 中。如果使用了 RESUEM NO，而且该 TABLESPACE 是非空的，并且没有使用 REPLACE NO，作业中断并报 8。对于非 SEGMENT 的 TABLESPACE，由于表空间无法重用，最好用 REPLACE 参数提高性能。

YES：将数据 LOAD 到一个非空的 TABLESPACE 中。如果 TABLESPACE 为空，会出现警告信息，但不会出错。数据以追加方式从当前的数据结尾处插入 TABLESPACE 中。被标记为 DELETE 的行及被删除的表所占用的空间不会被重用。

- **SHRLEVEL**：指定 LOAD Utility 操作时，其他的应用进程是否能访问该 TABLESPACE。

 NONE：默认值。指定做 LOAD 时没有并行应用进程访问该 TABLESPACE 或 PARTITION。

 CHANGE：指定在 LOAD 时可以有其他的应用进程对该 TABLESPACE 进行读写。注意 SHRLEVEL CHANGE 和下列参数是不能共存的：

 INCURSOR，RESUME NO，REPLACE，KEEPDICTIONARY，LOG NO，ENFORCE NO，SORTKEYS，STATISTICS，COPYDDN，RECOVERYDDN，PREFORMAT，REUSE，PART Integer REPLACE 等。

 使用了 LOAD SHRLEVEL CHANGE 将不会执行 SORT，BUILD，SORTBLD，INDEXVAL，ENFORCE 或 REPORT 阶段，并且需要考虑的兼容性方面的事情也不太一样。

 注意当使用了 SHRLEVEL CHANGE 时，TRIGGER 还保持有效，而使用 SHRLEVEL NONE 时则相反。

- **REPLACE**：指定在 LOAD 数据之前是否将 TABLESPACE 和 INDEX 清空。如果指定此参数，TABLESPACE 或 INDEX 中原有的数据将被新 LOAD 的数据覆盖，而不只是覆盖 LOAD 的那个 TABLE。对于被 DB2 STOGROUP 管理的文件，文件将被删除重建，除非同时指定了 REUSE 参数。不能将 REPLACE 和 PART integer REPLACE 一起使用，用户只能选择使用 REPLACE 覆盖整个 TABLESPACE 中的数据，或者使用 PART integer REPLACE 覆盖某个 PARTITION 的数据。

使用 LOAD REPLACE（而非 LOAD PART integer REPLACE）是对整个 TABLESPACE 层面的操作，这将会制止在 TABLESPACE 上不同 PARTITION 的并行操作。

- **STATISTICS：**收集 TABLESPACE、INDEX 的信息并更新 CATALOG 表，相当于做 RUNSTATS。
- **COPYDDN：**指定 INLINE COPY 的输出文件名。默认为 SYSCOPY。
- **REUSE ：**当使用了 REPLACE 选项时，LOAD Utility 可以重用 DB2-MANAGED 文件而不需要删除重建这些文件。REUSE 参数必须和 REPLACE 一起使用来重用文件。也可以对某个 PARTITION 使用 LOAD REPLACE 和 REUSE，这时只有该 PARTITION 对应的文件是可以被重用的。如果文件是经过多次扩展的，使用 REUSE 参数也不能释放这些扩展次数。
- **LOG：**在 RELOAD 阶段是否记 LOG。

 YES：默认值。所有 LOAD 的记录都记 LOG。

 NO：指定在 LOAD 过程中不记 LOG。LOAD Utility 完成后，TABLESPACE 上会有 COPY PENDING 状态。

 NOCOPYPEND：指定当 LOAD 完成后不在 TABLESPACE 上置 COPY PENDING 状态。注意，只有在确保 LOAD 的数据可以从其他地方恢复的时候才可以用这个参数。
- **SORTKEYS integer：**在 SORTBLD 阶段用并行的方式对 INDEX 的 KEY 进行排序。integer 为用户预估的 KEYS 的数量，默认值为 0。这样可以大大提升性能。一般情况下如果表有 INDEX，并且 UNLOAD 出来的数据不是按照 INDEX KEY 的顺序排放的，最好使用这个参数。
- **ERRDDN：**指定错误输出的文件，默认为 SYSERR。
- **MAPDDN：**用来存放 INPUT 文件中不合法的记录与表中的行的映射关系，默认为 SYSMAP。
- **DISCARDDN ddname：**存放由于 REFERENTIAL CONSTRAINT 或因为重复键值被删除的数据，默认文件名为 SYSDISC。
- **SORTNUM integer：**指定在做 SORT 时分配的临时文件的数量。
- **INTO-TABLE-SPEC：**可以在同一个 LOAD 作业中对多个表或多个 PARTITION 做 LOAD。至少要有一条 INTO TABLE 语句指定需要 LOAD 的表名。所有用 INTO TABLE 指定的表必须在同一个 TABLESPACE 中。

4.2.5.3　如何运行LOAD Utility

1. 运行之前的准备工作

UNLOAD 下来的数据是没有经过排序的，用户可以在做 LOAD 之前将 LOAD 的数据进行排序。

最好进行下面的检查：

- 确信在 UNIQUE INDEX 上没有重复键值；
- 预先更正或删除输入数据中违反 CHECK CONSTRAINT 和关系完整性的记录。

2. 作业中需要分配的文件

如表 4.6 所示。

表 4.6　作业中需要分配的文件

DATA SET	描　　述	是否必须
SYSIN	存放控制语句的文件	是
SYSPRINT	输出信息	是
INPUT DATASET	LOAD 数据所在的文件	是
SORT DATA SET	2 个用于排序的工作文件，分别为 SYSUT1（用于排序的输入）和 SORTOUT（用于排序的输出）	TABLE 上有 INDEX； 使用了 ENFORCE(CONSTRAINT)选项
MAPPING DATA SET	用于存放错误记录和对应表中的记录的映射关系	使用了 ENFORCE (CONSTRAINT)选项；表上有 UNIQUE INDEX 并且使用了 DISCARD 语句
UTPRINT	DFSORT 的输出信息，通常为 SYSOUT 或 DUMMY	如果使用了 DISCARD 语句
DISCARD DATA SET	存放由于错误被丢弃的记录。文件格式与输入的数据文件相同。默认文件名为 SYSDISC	如果使用了 DISCARD 语句
ERROR DATA SET	存放错误信息的文件	是
COPY DATA SET	INLINE IMAGE COPY 的输出备份文件	否

3. LOAD 变长类型的数据

要 LOAD 变长类型的数据，首先要在变长类型的数据前面加上 2 个字节的表示字段的长度。长度根据字段的类型来设置：

- 对于 VARCHAR 为单字节字符；
- 对于 VARGRAPHIC 为双字节字符。

例如对于一个变长的记录 X'42C142C142C2'，它可能是 VARCHAR 或 VARGRAPHIC

类型。如果是 VARCHAR 类型的，把它变为 X’0006’ X'42C142C142C2'；如果是 VARGRAPHIC 类型的，把它变为 X’0003’ X'42C142C142C2'。

4. 在 LOAD 中使用 REPLACE

- 我们可以对存放单个表和多个表的 TABLESPACE 使用 LOAD REPLACE。对于存放多个表的 TABLESPACE，使用 LOAD REPLACE 会清空 TBLESPACE 中所有的数据，而不是单独的一个表的数据。如果只要求覆盖某个单独表的数据，需要先删除该表中的所有数据，再使用 LOAD RESUME YES 以追加方式 LOAD 数据。
- 使用 LOAD REPLACE 控制语句而没有输入数据，这是一种很好的清空 TABLESPACE 的方法。我们在 JCL 中必须将数据文件 SYSREC 指定为 DUMMY。这种方法之所以有效体现在：
 — LOAD REPLACE 不需要将任何记录记入 LOG。
 — 重新定义 TABLESPACE 文件。
 — 保留 TABLESPACE 上所有的 VIEW 和相关的表的权限记录。
 — 可以使用 LOG YES 使之可以用 RECOVERY 恢复。
- 若 OBJECT 处于 REORG PENDING 状态，使用 LOAD REPLACE 可以清除这种 PENDING 状态。

5. 对 PARTITION 做 LOAD

使用 PART 子句可以对表的某几个 PART 进行 LOAD 操作，这时 REPLACE 和 RESUME 等选项也都对不同的 PARTITION 起作用。

注意：对于不同的 PARTITION，LOAD 控制语句要分开写，例如下面的语句。

```
  LOAD DATA  INTO TABLE DSN8710.EMP PART 1 REPLACE WHEN (1) = '0'
   ( EMPNO    POSITION (1:6)  CHAR(6),
     FIRSTNME  POSITION (7:18) CHAR(12),
……
                                  )
   INTO TABLE DSN8710.EMP PART 2 RESUME YES WHEN (1) = '1'
   ( EMPNO    POSITION (1:6)  CHAR(6),
     FIRSTNME  POSITION (7:18) CHAR(12),
 ……
                                     )
```

6. 关于提升 LOAD Utility 的性能

关于提升 LOAD UTIITY 的性能的几个建议如下。

- 当 LOAD 同一个 TABLESPACE 的多个表时，使用一条 LOAD DATA 语句，并在后面加上多条 INTO TABLE WHEN 子句。
- 对 PARTITION TABLESPACE 的不同的 PARTITION，用并行的 LOAD 作业进行 LOAD。
- 如果 TABLESPACE 上有一个或多个 NPI，那么必须在同一个作业中进行 LOAD；如果没有 NPI，那么可以在多个不同的作业中进行并行 LOAD。
- 对输入数据进行预处理，如重复键值的检查等。
- 避免数据类型转换，例如 integer 到 decimal 和 decimal 到 floating-point。
- 当使用 LOAD REPLACE 时，带 LOG NO 和 COPYDDN 参数做 INLINE COPY。
- 按照 CLUSTER INDEX 对数据进行排序，防止在完成 LOAD 后对 TABLESPACE 做 REORG。
- 避免进行排序。当输入数据存在下列几种情况时，排序不再进行：
 — 每个表都没有 KEY（INDEX KEY 和 FOREIGN KEY）。
 — 所有的 KEY 是同一类型。
 — 输入数据已经按照索引键值排序。
 — 输入的数据已经按照不同的表分组，并且每条记录只被 LOAD 到一个表中。
- 如果不能避免排序，使用 SORTKEYS 参数提高 SORT 的性能。
- 当 LOAD 的数据包括多个表的时候，使用 INTO TABLE WHEN 语句，将各表的数据分开 LOAD。

7. 使用 SORTKEYS

在使用 SORTKEYS 时，INDEX 被写到内存中而不是工作文件中，减少了 I/O，同时减少了 DASD 用来分配 SYSUT1 和 SYSOUT 的空间，尤其是作业提供了预估的要 SORT 的 KEYS 的数量。

如果 INDEX KEYS 是已经排过序的，就不需要使用 SORTKEYS 参数。

估算 KEYS 的数量的算法为：

- 对每个 INDEX 记为 1；
- 对每个非 INDEX 的 FOREIGN KEYS 记为 1；

- 对于同时为 INDEX 和 FOREIGN KEY 的记录，为第一个 RELATIONSHIP 记 0，其后的 RELATIONSHIP 记 1。
- 将该数量与行数相乘。

4.2.5.4 参考作业流

1. 使用 RESUME YES 和 ENFORCE NO 执行 LOAD

```
 //STEP1    EXEC DSNUPROC,UID='IUIQU2UB.LOAD',
 //         UTPROC='',
 //         SYSTEM='V71A'
//SYSREC   DD DSN=IUIQU2UB.LOAD.DATA,DISP=SHR,VOL=SER=SCR03,
 //         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
 //SYSUT1   DD DSN=IUIQU2UB.LOAD.STEP1.SYSUT1,
 //         DISP=(MOD,DELETE,CATLG),
 //         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
 //SORTOUT  DD DSN=IUIQU2UB.LOAD.STEP1.SORTOUT,
 //         DISP=(MOD,DELETE,CATLG),
 //         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
 //SYSIN    DD *
  LOAD DATA INDDN(SYSREC) RESUME YES
       INTO TABLE DSN8710.ACT
           (ACTNO    POSITION( 1) INTEGER EXTERNAL(3),
            ACTKWD   POSITION( 5) CHAR(6),
            ACTDESC  POSITION(13) VARCHAR)
       ENFORCE NO
 //*
```

2. 向一张表中 LOAD 数据

```
LOAD DATA INDDN(SYSREC) INTO TABLE NGDBA.TESTTB
```

3. 向两张表中分别 LOAD 数据

```
LOAD DATA INDDN(SYSREC) INTO TABLE NGDBA.TESTTB
                    INTO TABLE NGDBA.TESTTB2
```

4. 向表中 LOAD 选择的数据

```
  LOAD DATA RESUME YES
  INTO TABLE DSN8710.DEPT WHEN (1:3)='LKA'
    (DEPTNO POSITION (7:9) CHAR,
     DEPTNAME POSITION (10:35) CHAR,
     MGRNO POSITION (36:41) CHAR,
     ADMRDEPT POSITION (42:44) CHAR)
```

4.2.6　REBUILD INDEX Utility

4.2.6.1　Utility功能简介

REBULID INDEX 用来根据 TABLE 的内容重建 INDEX。

【运行阶段】如表 4.6 所示。

表 4.6　运行阶段及描述

阶　段	描　述
UTILINIT	初始化 Utility
UNLOAD	UNLOAD INDEX ENTRIES
SORT	将 UNLOAD 的 INDEX ENTRIES 进行排序
BUILD	重建 INDEX
SORTBLD	如果在控制语句中指定了 SORTKEYS 的选项，将进行并行的 BUILD 操作，原来的 SORT 和 BUILD 两步合并为 SORTBLD
UTILTERM	结束

4.2.6.2　控制语句及语法

【语法】请参考《DB2 UDB for z/OS V8 Utility Guide and Reference》。

【主要控制语句描述】

- **(index-name, ...)：**进行重建的 INDEX 的名称。格式为 creatorid.indexname。（All）表示对 TABLESPACE 下的所有 IDNEX 进行 REBUILD。
- **TABLESPACE database-name.table-space-name：**指定包含需要 REBUILD 的 INDEX 的 TABLESPACE 的名称。
- **PART integer：**指定 PARTITION INDEX 的 Physical PARITION 或 NON-PARTITION INDEX 的 Logical PARTITION 的序号。
- **SORTKEYS：**对 INDEX 进行并行的 REBUILD 以提升性能。
- **SORTNUM integer：**指定 SORT 过程中用到的临时文件的数量。
- **STATISTICS：**对执行 REBUILD 的 INDEX 同步收集系统方面的信息并写到 CATALOG 表。

4.2.6.3　运行REBUILD INDEX中需要注意的问题

1. 关于 PARALLLEL REBUILD INDEX

使用并行 REBUILD INDEX 可以并行地对 IDNEX KEYS 进行排序，并且可同时对多

个 IDNEX 进行 REBUILD。对于每个 INDEX，都有一对 SUBTASK 进行操作，一个负责对抽取出的 INDEX KEYS 排序，另一个负责 BUILD INDEX。如果带了 STATISTICS 参数，就有第三个 SUBTASK 负责收集 INDEX KEYS 的相关数据，并更新 CATALOG 表。

以下的几种情况用 PARALLEL REBUILD 对减少运行时间是最有效的：

- 一个 TABLESPACE 上有多个 IDNEX；
- PARTITIONED TABLESPACE 上的 PARTITIONING INDEX 或者 DATA-PARTITIONED INDEX；
- PARTITIONED TABLESAPCE 上的 NON-PARTITION INDEX。

从 DB2 V8 开始，REBUILD INDEX 将自动对 PARTITIONED TABLESAPCE 采取并行 REBUILD 的方式。用户可让 Utility 自动分配 SORT 空间进行 REBUILD，也可采取下列三种方法之一来分配 SORT WORK DATASET 和 MESSAGE DATASET：

方法 1：由 DB2 动态地分配所需的文件

- 在作业控制语句中加上 SORTKEYS 和 SORTDEVT 参数；
- 不在作业中指定 SORTWKnn 文件，由系统动态地进行分配；
- 将 UTPRINT 分配到 SYSOUT。

方法 2：由用户分配 SORT WORK 文件，系统动态地分配 MESSAGE 文件

- 在作业控制语句中指定 SORTKEYS 参数；
- 在 JCL 中指定 SWnnWKnn 格式的文件名；
- 分配 UTPRINT 到 SYSOUT。

方法 3：由用户在作业中显式地分配 SORT WORK 和 MESSAGE 文件

- 在作业控制语句中指定 SORTKEYS 参数；
- 在 JCL 中指定 SWnnWKnn 格式的文件名；
- 指定格式为 UTPRINnn 格式的文件名。

每个 SORT SUBTASK 都会使用一对 SORT WORK 文件和 MESSAGE 文件。另外，当 REBUILD PARTITION TABLESPACE 中的 NON-PARTITIONING INDEX 时，DB2 需要为其分配一个单独的 MERGE MESSAGE 文件。

DDNAME SWnnWKmm：用来做排序时的 WORK 文件。nn 描述了 SUBTASK 的序号；mm 描述了 SUBTASK 中的文件的序号。DDNAME UTPRINnn：用来做排序的输出信息文件。nn 描述了 SUBTASK 的序号。

REBUILD IDNEX 运行过程中的 SUBTASK 的数量是由下面的几个因素决定的：

- SUBTASK 的数量等于 WORK 文件的数量；
- SUBTASK 的数量等于分配的 MESSAGE 文件的数量；
- 如果用户同时定义了 WORK 文件和 MESSAGE 文件，SUBTASK 的数量等于其中较小的数量。

2. ESET REBUILD-PENDING 状态

REBUILD-PENDING 状态表示 INDEX 处于 REBUILD-PENDING 状态。

共有如下三种不同的情况。

- RBDP：PARTITIONING INDEX 的 PHYSICAL 或 LOGICAL PARTITION 处于 REBUILD-PENDING 状态。只有单独的 INDEX PARTITION 不可被访问。通过 REBUILD 这些单独的 PARITITION 可以 RESET REBUILD-PENDING 状态。
- RBDP*：NON-PARTITIONING INDEX 处于 REBUILD-PENDING 状态。整个 INDEX 都无法访问。只要 REBUILD 受影响的 LOGICAL PARTITION，就可以 RESET REBUILD-PEDING 状态。
- PSRBD：NON-PARTITIONING INDEX SPACE 处于 REBUILD-PENDING 状态，整个 IDNEX SPACE 不可用，需要对整个 IDNEX 做 REBUILD。

用下面几种方法 RESET REBUILD-PENDING 状态：

- REBUILD INDEX
- REORG INDEX SORTDATA
- REPAIR SET IDNEX WITH NORBDPEND
- -START DATABASE(XXXX) SP(XXXX) WITH ACCESS(FORCE)

4.2.6.4　参考作业流

1. REBUILD A INDEX

```
//REBUILD JOB (ACCTNUM,EXP),'PGMRNAME',
//        TIME=1440,
//        NOTIFY=&SYSUID,
//        CLASS=A,
//        MSGCLASS=X,
//        MSGLEVEL=(1,1)
//STEP1   EXEC PGM=DSNUTILB,PARM='PB01,RBIX',REGION=0M
//STEPLIB  DD  DSN=DSN810.SDSNLOAD,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//UTPRINT  DD  SYSOUT=*
```

```
//SYSIN    DD  *
   REBUILD INDEX(SYSIBM.DSNKDX03)
```

2. REBUILD INDEX PARTITIONS

```
REBUILD INDEX (DSN8810.XEMP1 PART 2, DSN8810.XEMP1 PART 3)
```

3. REBUILD INDEX ALL 带 SORTKEYS

```
REBUILD INDEX(ALL) TABLESPACE TESTDB.TESTTS
SORTKEYS SORTDEVT SYSDA REUSE
```

4.2.7 RECOVER Utility

4.2.7.1 Utility功能简介

该 Utility 将数据恢复到当前状态或以前的某个时间点，然后通过 APPLY LOG，将 LOG 中的记录应用到 TABLESPACE 的数据中。RECOVER 的对象最大的是 TABLESPACE 或 INDEXSPACE，最小的是 PAGE。对于 TABLESPACE，可以恢复整个 TABLESPACE 或某个 PARTITION，也可以是一段连续的 PAGE 或某个单独的 PAGE。

如果 RECOVER 指定的 FULL IMAGE COPY 文件丢失，DB2 会向前寻找最近一个可用的 FULL IMAGECOPY 文件来恢复当前的 TABLESPACE 或 INDEXSPACE。

【输出】当使用了 TOLOGPOINT、TORBA、TOCOPY 等用于恢复到某个时间点的控制语句时，RECOVER 会将 INDEX 置为 REBUILD-PENDING 的状态。用户需要在 RECOVER 执行后再运行 REBUILD INDEX 作业来清除 IDNEX 的 REBUILD-PENDING 状态。如果对有关联关系的表所在的 TABLESPACE 进行 RECOVER，务必对整个 TABLESPACE 进行 RECOVER，以及对所有的 INDEX 进行 REBUILD，否则 DB2 会在 TABLESPACE 上置 CHECK-PENDING 状态。

【运行阶段】如表 4.7 所示。

表 4.7 运行阶段及描述

阶　段	描　述
UTILINIT	初始化
RESTORE	定位、合并相关的 IMAGECOPY 文件，将 TABLESPACE 恢复到备份的文件；如果指定了 PARALLEL 选项，则对多个 OBJECT 进行并行处理
RESTORER	如果指定了 PARALLEL，读取并合并 IMAGECOPY 文件
RESTOREW	如果指定了 PARALLEL，将备份内容写到 OBJECT

续表

阶　段	描　述
LOGAPPLY	将前几个步骤执行时的 LOG 的变化反映到目标中
UTILTERM	结束

4.2.7.2　控制语句及语法

【语法】请参考《DB2 UDB for z/OS V8 Utility Guide and Reference》。

【主要控制语句描述】

- **LIST** listdef-name：指定包含 OBJECT 的 LIST 的名称。
- **TABLESPACE** database-name.table-space-name：指定要恢复的 TABLESPACE 的名称。
- **INDEXSPACE** database-name.index-space-name：指定要恢复的 INDEXSPACE 的名称。
- **INDEX** creator-id.index-name：指定要恢复的 INDEX 的名称。

以下参数为可选的。

- **DSNUM**：指定恢复 TABLESPACE 或 INDEXSPACE 的某个或某几个 PART。

 ALL：默认值，恢复所有的 PART。

 INTEGER：指定要恢复的 PARTITION 的数量。最大值为 254。
- **PAGE page-number**：指定要恢复的 PAGE。

 PAGE 参数不能和 LIST 连用。

 Page-number：page 的地址，即序号，可以用 DECIMAL 或 16 进制数表示。
- **CONTINUE**：指定 RECOVERY 操作将要继续。当 RECOVER 操作在重建一个 PAGE 时被异常中止，该 PGAE 被标记为“ERROR”状态。使用该选项可在上一次恢复之后，从失败点开始重新对该 PAGE 进行恢复。
- **TORBA X'byte-string'**：12 位的 16 进制数，用于 NON-DATASHARING 环境，指定 OBJECT 恢复到 LOG 的某个 RBA 点。
- **TOLOGPOINT X'byte-string'**：指定 OBJECT 被恢复到 LOG 的某个 RBA 点或 LRSN 点。DB2 将根据 SYSIBMS.SYCOPY 中的记录找到最近的 FULL IMAGE COPY 文件进行恢复，然后根据 LOG 的记录恢复到某个 LOG 点。
- **LOGONLY**：将目标 OBJECT 从已经存在的文件（比如用 DFSMS 进行备份的），根据 LOG 的记录恢复到某个 LOG 点。

- **REUSE**：指定 RECOVER Utility 可以重用 DB2-MANAGED 的文件，而不需要删除并重新定义。如果不指定 REUSE 参数，RECOVER Utility 将删除并重新定义文件。
- **TOCOPY data-set-name**：恢复到一个指定的 IMAGECOPY 文件。若该文件是一个 FULL IMAGE COPY 文件，只要用它来做恢复就可以了；如果它是一个 INCREMENTAL IMAGECOPY 文件，那么还要用到之前的一个 FULL IMAGECOPY 文件，以及中间若干个 INCREMENTAL IMAGECOPY 文件。

 如果使用 TOCOPY 或 TORBA 恢复某个 NON-PARTITION TABLESPACE 对应实体文件中的某一个单独的文件，DB2 会报出 DSNU520I 的警告信息，表示 TABLESPACE 在恢复结束后可能会处于不一致的状态。如果使用 TOCOPY 恢复 PARTITION TABLESPACE 的某个 PARTITION，则必须使用该 PARTITION 对应的 IMAGECOPY 文件或整个 TABLESPACE 的 IMAGECOPY 文件。如果控制语句中指定了 LIST，那么 TOCOPY 无效。
- **TOLASTCOPY**：恢复到最后一个 COPY 文件。如果最后一个 COPY 文件是 FULL IMAGE COPY，则恢复到该 FULL IMAGE COPY 文件；如果最后一个 COPY 文件是 INCREMENTAL IMAGECOPY 文件，则从包括最近的一次 FULL IMAGE COPY 文件开始以来的所有 INCREMENTAL IMAGECOPY 文件一起恢复。
- **TOLASTFULLCOPY**：只恢复到最近的 FULL IMAGE COPY 文件。
- **ERROR RANGE**：指定所有 REPORT 有 I/O 错误的 PAGE 被恢复。

4.2.7.3 如何运行RECOVER Utility

1. RECOVER 数据表和索引

没有必要同时恢复数据和 INDEX 的内容，在下面几种情况下，需要 REBUILD INDEX：

- TABLESPACE 被恢复到某个时间点；
- INDEX 被损坏；
- INDEX 处于 REBUILD-PENDING 状态；
- 该 INDEX 没有可用的 IMAGECOPY 文件。

当用户需要同时恢复数据和 INDEX，而 INDEX 没有可用的 COPY 文件时，可以采用下面的方法：

- 用 RECOVER TABLESPACE 恢复数据；
- 用 REBUILD INDEX 根据数据重建 INDEX。

如果同时有 TABLESPACE 和 INDEX 的 COPY 文件，可以在一条 RECOVER 控制语句中同时恢复 DATA 和 INDEX。

2. 恢复一个或多个 TABLESPACE

- 可以用一条 RECOVER 控制语句恢复一个 TABLESPACE，例如：

```
RECOVER TABLESPACE TESTDB.TESTTS1
```

- 也可以在一条 RECOVER 控制语句中恢复多个 TABLESPACE，例如：

```
RECOVER TABLESPACE TESTDB.TESTTS1 DSNUM 2
TABLESPACE TESTDB.TESTTS2 TORBA X'000007425468'
TABLESPACE TESTDB.TESTTS3
```

- 最大并行数取决于可以获取的 DEVICE 数量。这个数值可以用 TAPEUNIT 参数控制。

3. 恢复一个 DATASET 或一个 PARTITION

用 RECOVER Utility 可以恢复 PARTITION TABLESPACE 的某个 PARTITION 或 DATASET。如果备份 TABLESPACE 时是 DATASET 级别的，那么恢复时也必须是 DATASET 级别的。而备份 TABLESPACE 时是整个 TABLESPACE 级别的，则恢复时可以选择恢复整个 TABLESPACE 或恢复某个单独的 PARTITION 或 DATASET（使用 DSNUM 参数）。

4. 恢复一个 PAGE

使用 RECOVER Utility 可以恢复一个数据被损坏的 PAGE。使用 RECOVER PAGE 和 CONTINUE 可以恢复一个在 LOGAPPLY 阶段被损坏的 PAGE。首先从 DB2 的错误信息中获得数据损坏的 PAGE 的地址。在运行过程中信息 DSNI012I 将报告某个 PAGE 被损坏；运行结束时，信息 DSNU501I 也会报告某个 PAGE 被损坏。

恢复步骤如下：

- 使用 REPAIR Utility 的 DUMP 选项查看被损坏的 PAGE 的内容，根据 LOG APPLY RECORD 确定被修改的值，并使用 REPAIR 的 REPLACE 选项将该值 APPLY 到 PAGE 中；然后使用 REPAIR 的 RESET 关闭 PAGE 上的不一致状态。注意，使用 REPAIR 必须十分小心，特别是对 HEADER PAGE 使用 REPLACE 时。操作不当时会造成数据丢失甚至系统错误。
- 重新提交 RECOVER 作业，RECOVER TABLESPACE *DBNAME.TSNAME* PAGE（*PAGE_NUM*）CONTINUE。
- 如果有多个损坏的页，对每个 PAGE 重复上两个步骤。

4.2.7.4 参考作业流

1. RECOVER TABLESPACE

```
RECOVER  TABLESPACE NASEDB.NSCHDTL
```

2. RECOVER TABLESPACE 的某个分区

```
RECOVER  TABLESPACE NASEDB.NSCHDTL DSNUM 2
```

3. 将 TABLESPACE RECOVER 到指定的 RBA

```
RECOVER TABLESPACE TESTDB.TESTTS6 TO RBA X'BA73F65B2911'
```

4. 将若干 TABLESPACE RECOVER 到指定的时间点

```
RECOVER TOLOGPOINT X'BB67EFE541C4' PARALLEL(4)
     TABLESPACE TEMPDB.TEMPTS
     TABLESPACE TESTDB.TESTTS1
     TABLESPACE TESTDB.TESTTS2
     TABLESPACE TESTDB.TESTTS3
```

5. RECOVER 一个 INDEXSPACE

```
RECOVER INDEXSPACE TEMPDB.TEMPIX0 DSNUM 1 TOLASTCOPY
```

4.2.8 REORG TABLESPACE Utility

4.2.8.1 Utility功能简介

REORG TABLESPACE Utility 对 TABLE SPACE 进行重组，以提升访问 TABLESPACE 的性能，并回收碎片。在重组时，可以通过指定 STATISTICS 参数，进行收集与访问路径相关数数据，存放在 CATALOG 表中。

【运行阶段】如表 4.8 所示。

表 4.8 运行阶段及描述

阶　段	描　述
UTILINIT	启动，初始化
UNLOAD	UNLOAD TABLESPACE 中的数据；如果有 CLUSTERING INDEX，并且控制语句中有 SORTDATA 或 SHRELVEL，则要对数据进行排序。如果加了 NOSYSREC 参数，则将数据送入内存，在 RELOAD 阶段使用；否则，UNLOAD 到 PS 文件中

续表

阶　段	描　述
RELOAD	从 PS 文件中将数据 RELOAD 到 TABLESPACE 中。若指定 COPYDDN，SHRLEVEL REFERENCE 或 CHANGE，对 TABLESPACE 做 COPY。若加了 SORTKEYS 参数，则多一个 SUBTASK 对 INDEX KEYS 进行排序。更新 TABLESPACE 和 TABLE 的 STATISTICS
SORT	对 INDEX KEYS 进行排序。如果加了 SORTKEYS 参数，则将排序的 INDEX KEYS 送到内存中，在 BUILD 阶段使用
BUILD	BUILD INDEX，更新 INDEX STATISTICS
SORTBLD	如果指定了 SORTKEYS 参数，进行并行的 INDEX BUILD，所有在 SORT 和 BUILD 阶段做的动作都在这个阶段完成
LOG	进行交互式的 LOG PROCESS。只有在 SHRLEVEL CHANGE 时才会用到
SWITCH	将对 TABLESPACE 的访问切换到 TABLESPACE 或 PARTITION 的 SHADOW 文件
BUILD2	更新 NPI（如果 TABLESPACE 上有 NPI 且控制语句中用了 REORG TABLESPACE PART n SHRLEVEL REFERENCE 或 CHANGE）
UTILTERM	结束

4.2.8.2　控制语句及语法

【语法】请参考《DB2 UDB for z/OS V8 Utility Guide and Reference》。

【主要控制语句描述】

- **TABLESPACE database-name.table-space-name**：需要做 REORG 的 TABLESPACE 的名称。
- **REUSE**：当使用 SHRLEVEL NONE 时起作用。DB2 REORG 重用原来的 DB2-MANAGEED DATASET，而不需要删除重建。如果文件有多次扩展，则 REORG 以后文件的扩展次数不会被释放。此参数在 SHRLEVEL REFERENCE 和 CHANGE 时不起作用。
- **PART integer，PART integer1:integer2**：对 TABLESPACE 的某个或某几个 PARTITION 做 REORG。Integer 的最大值为 254。

 Integer：指定一个单独的 PARTITION。

 Integer1:integer2：指定一个 PARTITION 的范围。
- **LOG**：指定在 RELOAD 阶段是否记 LOG。如果不记 LOG，TABLESPACE 需要做 IMAGECOPY。默认值为 YES。
- **SORTDATA**：指定数据在 UNLOAD 时使用 TABLESPACE SCAN，然后按 CLUSTERING

INDEX 进行排序。推荐使用该参数以提升性能，除了以下情况之一：

— 数据完全按照 CLUSTER 的顺序排放，并且 REORG UTILTIY 用来回收删除的表的空间；

— TABLESPACE 非常大，而且没有足够的空间用来做 SORT；

— 最长的记录长度超过 32 760。

- **NOSYSREC**：指定 SORT 的输出直接作为 RELOAD 的输入，而不需要 UNLOAD DATASET。只有在使用 REORG TABLESPACE， SORTDATA， SHRLEVEL REFERENCE 或 NONE，并且没有使用 UNLOAD PAUSE 和 UNLOAD ONLY 时，才可以使用 NOSYSREC 参数。
- **SORTKEYS**：INDEX KEYS 在 SORTBLD 阶段使用并行方式进行 SORT 和 BUILD。推荐使用该参数以提升 REORG 的性能。
- **COPYDDN ddname1，ddname2**：指定 IMAGECOPY 时所使用的 PRIMARY(ddname1)和 BACKUP(ddname2)备份文件名称。PRIMARY 默认为 SYSCOPY。
- **SHRLEVEL**：指定 REORG 过程中应用进程对 OBJECT 的访问方式。

 NONE：在 REORG 的 UNLOAD 阶段，应用程序对被 REORG 的区域可以读但不能写；RELOAD 阶段，应用程序不能进行读写操作。默认值为 NONE。

 REFERENCE：REORG 在进行 UNLOAD 数据时，应用程序对正在 REORG 的区域可以读但不能写。在将数据 RELOAD 到 SHADOW COPY 文件时，此时应用程序可以对原 TABLESPACE 文件执行读操作但是不可以对原 TABLESPACE 文件进行写操作。在 SWITCH 阶段，不允许读写。随后应用程序可以正常地进行读写。SHRLEVEL REFERENCE 参数和下列参数不能并存：LOG，UNLOAD，MAPPINGTABLE，MAXRO，LONGLOG。

 CHANGE：REORG 的 UNLOAD 阶段，应用程序可以对 REORG 的区域进行读写操作。对 SHADOW COPY 文件进行 RELOAD 时，应用程序可以对原文件进行读写操作。根据 LOG 将原文件中的修改内容补充到 SHADOW COPY 文件中，此时应用程序可以读，并且在大部分时间内可以写原文件（除了补充 LOG 记录的最后一小部分时间）。在 SWITCH 阶段不能进行读写，然后可以恢复正常的读写操作。SHRLEVEL CHANGE 与下列参数不能并存：LOG，SORTDATA，NOSYSREC，SORTKEYS，UNLOAD。
- **DRAIN_WAIT integer**：指定 DRAIN 操作时，REORG 等待 DML 操作（INSERT，

DELETE，UPDATE，SELECT）的时间，单位为秒，取值范围 0～1 800 秒。该参数会覆盖 DSNZPARM 中的 IRLMWT 和 UTIMOUT，但只针对 DML 操作。对于其他的 TASK，如 COMMAND 等，还是 IRLMWT 和 UTIMOUT 起作用。如果指定 0，则使用 IRLMWT 和 UTIMOUT。

- **RETRY integer**：REORG 尝试做 DRAIN 操作的最大次数，取值范围为 0～255。如果忽略该参数，则不会有 RETRY。需要注意的是，使用 RETRY 参数会增加一定的系统开销，并使只读的时间延长。
- **MAPPINGTABLE table-name**：指定 MAPPING TABLE 的名称。MAPPING TABLE 用来映射原 TABLESPACE 文件中的记录 RID 和 SHADOW COPY 文件中的对应关系。在使用 SHRLEVEL CHANGE 时必须在运行 REORG 之前新建一个 MAPPING TABLE 及其 INDEX。它的字段和索引都有特定的描述，用户必须对 MAPPING 有 DELETE，INSERT，SELECT，UPDATE 的权限。当并行地运行 REORG 作业时，需要分别使用不同的 MAPPING TABLE。以下为 MAPPING TABLE 及 INDEX 的定义：

```
CREATE TABLESPACE table-space-name SEGSIZE 64;
CREATE TABLE table-name1
   (TYPE       CHAR(1) NOT NULL,
    SOURCE_RID  CHAR(5) NOT NULL,
    TARGET_XRID CHAR(9) NOT NULL,
    LRSN       CHAR(6) NOT NULL);
  CREATE UNIQUE INDEX index-name1 ON table-name1
   (SOURCE_RID ASC, TYPE, TARGET_XRID, LRSN);
```

- **MAXRO integer**：指定 LOG PROCESSING 的最后一个 ITERATION 可以运行的最大时间。在此阶段，应用程序对数据是只读的。

注意：实际的运行时间可能超过这个时间。用 ALTER Utility 可以修改 MAXRO 的值。默认值为 300 秒。

integer：指定时间，单位为秒。设定一个较小的值有助于减少只读的时间，但这会增加 REORG 的运行的总时间。如果设定一个比较大的值，则下一个 LOG PROCESS ITERATION 很可能就是最后一个 ITERATION。

DEFER：表示 LOG APPLY 的 LAST ITERATIION 在什么时候结束是不确定的，由系统来估算这个时间，需要和 LONGLOG CONTINUE 参数一起使用。

- **DRAIN**：指定在 LOG PROCESS 的最后阶段，当 MAXRO 的值被突破，而且要进行 LOG PROCESSING 的 LAST ITERATION 时 DRAIN 的类型。

 WRITERS：默认值。当达到 MAXRO 的时候只 DRAIN 写操作，并在随后的 SWITH 阶段中 DRAIN 所有的操作。

 ALL：当达到 MAXRO 的 THREASHOLD 时，DRAIN 所有的读/写操作。考虑在下面的情况下设置 DRAIN ALL：

 — 在 LOG APPLY 阶段有大量的 UPDATE 操作；

 — 当默认选项可能会导致大量的-911（超时）错误时。

- **LONGLOG**：表示在 LOG PROCESS 时下一个 ITERATION 所处理的 RECORD 数量并不比上一个 ITERATION 所处理的 RECORD 数量少时 DB2 所采取的操作。这种情况表示 Utility 读 LOG 的操作赶不上应用程序写 LOG 的操作。

 CONTINUE：默认值，表示直到作业结束前，Utility 一直执行，包括 LOG PROCESSING。通常 MAXRO DEFER 和 LONGLOG CONTINUE 一起使用，表示 REORG 一直允许对正在 REORG 而且没有 SWITCH 到 SHADOW COPY 的数据的访问。

 TERM：表示在 DELAY 参数指定的时间后结束 Utility 的运行。

 DRAIN：表示在 DELAY 参数指定的时间后 DRAIN 所有的 WRITE 操作，并强制 LOG PROCESS 的最后的 ITERATION 发生。

- **DELAY integer**：单位为秒，默认时间为 1 200 秒。指定 ONLINE REORG 向控制台发出 LONGLOG 的信息和 REORG 执行 LONGLOG 所设置的操作之间的时间间隔。

- **TIMEOUT**：指定当 REORG 在 LOG APPLY 阶段或 SWITCH 阶段进行 DRAIN 操作发生超时的时候所采取的操作。

 ABEND：默认值。作业发生 ABEND，并使 OBJECT 处于 UTUT 或 UTRO 状态。

 TERM：当使用 TERM 时，DB2 会采用下列的操作：

 — 发出一个隐式的 TERM Utility 命令，使作业结束，返回为 8；

 — 输出 DSNU590I 和 DSNU170I 的信息；

 — OBJECT 处于 RW 状态。

- **UNLOAD**：指定 REORG 作业在做完 UNLOAD 操作后是否继续执行。

 CONTINUE：默认值。作业继续执行。

PAUSE: 作业中断，Utility 信息记录在 SYSIBM.SYSUTIL 中，可以使用 RELOAD RESTART(PHASE)重启。当用户希望重新定义文件时，该参数比较有用。操作流程如下：

— 运行 REORG UNLOAD PAUSE；

— 用 IDCAMS 重新定义文件；

— 通过指定 RESTART(PHASE)，重新启动 REORG 作业。

ONLY：作业结束，SYSIBM.SYSUTIL 中的记录被清除。

EXTERNAL：作业结束，SYSIBM.SYSUTIL 中的记录被清除，数据 UNLOAD 到外部文件。

- **STATISTICS**：指定对 INDEX 进行 STATISTICS 的收集。STATISTICS 被 REPORT 或存储到 DB2 CATALOG 表中。
- **UNLDDN ddname**：UNLOAD 数据文件，默认为 SYSREC。
- **WORKDDN(ddname1,ddname2)**：指定用于中间输出的临时文件。

 ddname1：用于数据排序输入的临时文件，默认为 SYSUT1；

 ddname2：用于数据排序输出的临时文件，默认为 SORTOUT。
- **SORTDEVT**：指定动态分配的临时文件的 DEVICE 类型。
- **SORTNUM integer**：指定动态分配临时文件的数量。必须和 SORDEVT 一起使用。

4.2.8.3　如何运行REORG TABLESPACE Utility

1. 运行 REORG 前需要注意的事项

- CATALOG 和 DIRECTORY TABLESPACE：在对 CATALOG 和 DIRECTORY TABLESPACE 文件做 REORG 之前，必须对它们做 IMAGECOPY。
- REGION SIZE：建议使用 4096KB。
- MAPPING TABLE 和 SHRLEVEL CHANGE：在使用 SHRLEVEL CHANGE 选项前，需要建立一个 MAPPING TABLE 及其 INDEX。
- TABLESPACE 上的几种情况禁止 REORG 的执行：
 — PARTITION TABLESPACE 的某个或某几个 PARTITION 处于 RECOVER-PENDING 状态；
 — CLUSTERED INDEX 处于 REBUILD-PENDING 状态，并且 UNLOAD 时要用到 CLUSTERED INDEX。

同样，在下面的几种情况下，不能对单独的 PARTITION 做 REORG：

— PARTITION 处于 RECOVER-PENDING 状态；

— PARTITION 对应的 PARTITIONING INDEX 处于 REBUILD-PENDING 或 RECOVER-PENDING 状态，并且数据 UNLOAD 时使用 CLUSTERED INDEX。

2. 什么时候需要做 REORG

TABLESPACE 或 PARTITION 处于 REORG-PENDING 状态时，必须做 REORG。用 -DISPLAY DB(dbname) SP(spname) RES 查看 TABLESPACE 的 RESTRICT 状态。

另外，可以通过直接查询 DB2 CATALOG 表 SYSTABLEPART 和 SYSINDEXPART 中相关字段的以确定哪些 TABLESPACE 或 INDEX 需要 REORG。这些信息对 CATALOG TABLESPACE 同样适用，除了 SYSDBASE，SYSVIEWS，SYSPLAN 等表。

3. 关于 SHRLEVEL 的使用

如果 REORG 时选择了 SHRLEVEL CHANGE 参数，DB2 通过对 LOG 的操作来更新 SHADOW 文件使其数据在 SWITHING 之前能够与原文件保持一致。LOG PROCESS 的第一个 ITERATION 对上一个 ITERATION 的积累的数据进行处理，如此循环，直到下列情况之一发生：

- DB2 对 LOG PROCESS 的下一个 ITERATION 所用的时间进行估算，如果这个时间小于或等于控制语句中指定的 MAXRO 值，那么下一个 ITERATION 将是 LOG PROCESS 的最后一个 ITERATION。
- DB2 估算 SWITCH 的时间，如果不能在指定的 DEADLINE 之前开始，则终止 REORG 操作。
- 当 LOG PROCESS 时下一个 ITERATION 所处理的 RECORD 数量并不比上一个 ITERATION 所处理的 RECORD 数量少时，如果上两个条件没有达到，则 DB2 发出 DSNU377I 的信息。DB2 在超出 DELAY 指定的时间之前继续进行 LOG PROCESS 的操作，然后执行 LONGLOG 所指定的操作。

4. 关于 REORG 的性能

- 建议使用 SORTDATA 选项，除非 TABLESPACE 中的数据量非常大，或者用户不希望占用额外的硬盘空间做 DFSORT。在此情况下，DB2 用 TABLESPACE SCAN UNLOAD 数据，并按照 CLUSTER INDEX 对数据进行排序。如果 TABLE 的 CLUSTERATIOF < 95%，或 FAROFFPOS / CARD > 5%，用 SORTDATA 的效果比较好。一般来说 CLUSTERATIOF 越低，用 SORTDATA 的效果就越好。

- 并行地对 PARTITION TABLESPACE 的不同 PARTITION 做 REORG。这样多个作业的 CPU 时间的总和会多过用单独一个作业对整个 TABLESPACE 做 REORG 的方式，但是整体执行时间会大大降低。
- 使用 SORTKEYS 选项在 SORT 和 BUILD 阶段对 KEYS 进行并行排序。用该选项后 INDEX KEYS 被写到内存中而不是外部文件中。另外由于不需要 SYSUT1 和 SORTOUT 文件，因而节省了 WORK 空间。当有多个 INDEX 需要 BUILD 时，用 SORTKEYS 的选项的性能会比较好。在使用 SHRLEVEL CHANGE 时，DB2 会自动使用 SORTKEYS 的选项。如果使用了 SORTKEYS 的 REORG 作业在 RELOAD，SORT，BUILD 或 SORTBLD 阶段 ABEND，则只能从 RELOAD 阶段重新开始。

5. 中止和重启 REORG Utility

如果在 UNLOAD 阶段中止作业（用 TERM Utility 的命令，以下同）的运行，OBJECT 并没有被修改，作业可以正常重启。

如果在 RELOAD 阶段中止作业的运行，则采取的操作要取决于 SHRLEVEL 选项：

- SHRLEVEL NONE：数据没有被修改，TABLESPACE 和 INDEXSPACE 处于 RECOVER-PENDING 状态。只要 RECOVER TABLESPACE 后就可以重新运行 REORG 作业。
- SHRLEVEL REFERENCE 或 CHANGE：数据被 RELOAD 到 SHADOW 文件，而原文件并没有受到影响，因此只要直接重新启动 REORG 就可以了。

如果在 SORT，BUILD 或 LOG APPLY 阶段中止作业的运行，所采取的操作要取决于 SHRLEVEL 选项：

- SHRLEVEL NONE：INDEX 未 BUILD，处于 RECOVER 状态，可以使用 SORTDATA 选项运行 REORG 或者使用 REUILD INDEX Utility。
- SHRLEVEL REFERENCE 或 CHANGE：数据被 RELOAD 到 SHADOW 文件，而原文件并没有受到影响，因此直接重新启动 REORG 即可。

如果在 SWITCH 阶段中止作业运行，所有改名到 SHADOW 文件名的文件被改回到原来的文件名，因此 OBJECT 仍处于原来的状态，可以重新启动作业。如果在改名回原文件的过程中发生错误，则 OBJECT 处于 RECOVER-PENDING 状态。如果 SWITCH 阶段没有完成，那么 INLINE COPY 所生成的 COPY 文件无法用于恢复。

如果在BUILD2阶段中止REORG Utility，LOGICAL PARTITION处于RECOVER-PENDING状态，只要对 INDEX 做 REBUILD 就可以完成 REORG。

4.2.8.4 参考作业流

1. 最简单的 REORG 作业，使用 SORTDATA 选项

```
//JOBLIB  DD  DISP=SHR,DSN=DSN710.SDSNLOAD
//PH01S18 EXEC DSNUPROC,PARM='ZB0B,REORG'
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR
//SORTOUT  DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SORTWK01 DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SORTWK02 DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SYSPRINT DD  SYSOUT=*
//UTPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=3390,SPACE=(CYL,(10,50),,,ROUND)
//SYSREC   DD  UNIT=3390,SPACE=(CYL,(10,50),,,ROUND)
//SYSIN    DD  *
REORG TABLESPACE TEMPDB.TEMPTS SHRLEVEL NONE
```

2. REORG 一个 PARTITOIN TABLESPACE 的 PARTITION

```
//JOBLIB  DD  DISP=SHR,DSN=DSN710.SDSNLOAD
//PH01S18 EXEC DSNUPROC,PARM='SB0A,REORG'
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR
//SORTOUT  DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SORTWK01 DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SORTWK02 DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SYSPRINT DD  SYSOUT=*
//UTPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=3390,SPACE=(CYL,(10,50),,,ROUND)
//SYSREC   DD  UNIT=3390,SPACE=(CYL,(10,50),,,ROUND)
//SYSIN    DD  *
REORG TABLESPACE NASEDB.NSPABRP PART 1:5 SHRLEVEL NONE
```

3. 使用 SORTKEYS 选项

```
//JOBLIB  DD  DISP=SHR,DSN=DSN710.SDSNLOAD
//PH01S18 EXEC DSNUPROC,PARM='SB0A,REORG'
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR
//SORTOUT  DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SW01WK01 DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SW01WK02 DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SW01WK03 DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SW02WK01 DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SW02WK02 DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SW02WK03 DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
```

```
//SW03WK01 DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SW03WK02 DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SW03WK03 DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SORTWK01 DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SORTWK02 DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SORTWK03 DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SYSPRINT DD  SYSOUT=*
//UTPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=3390,SPACE=(CYL,(10,50),,,ROUND)
//SYSREC   DD  UNIT=3390,SPACE=(CYL,(10,50),,,ROUND)
//SYSIN    DD  *
REORG TABLESPACE NASEDB.NSPABRP LOG NO SORTDATA SORTKEYS
```

4. 使用 SHRLEVLEL CHANGE，动态分配 SORTKEYS 所需的临时文件

```
//JOBLIB  DD  DISP=SHR,DSN=DSN710.SDSNLOAD
//PH01S18 EXEC DSNUPROC,PARM='SB0A,REORG'
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR
//SORTOUT  DD  UNIT=3390,SPACE=(CYL,(10,20),,,ROUND)
//SYSPRINT DD  SYSOUT=*
//UTPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=3390,SPACE=(CYL,(10,50),,,ROUND)
//SYSREC   DD  UNIT=3390,SPACE=(CYL,(10,50),,,ROUND)
//SYSCOPY  DD  UNIT=3390,SPACE=(CYL,(10,40),,,ROUND)
//SYSIN    DD  *
REORG TABLESPACE NASEDB.NSPABRP LOG NO SORTDEVT SYSDA  SORTNUM 3
   SHRLEVEL CHANGE MAPPINGTABLE MAPTAB
```

5. 使用 STATISTICS 选项

```
REORG TABLESPACE NASEDB.NSPABRP STATISTICS
```

4.2.9　REPAIR Utility

4.2.9.1　Utility功能简介

REPAIR Utility 可以帮助用户修复数据，特别是对于用户一般情况下无法访问的数据，例如 SPACE MAP PAGES 和 INDEX ENTRIES 等 DB2 控制数据。因此，必须很谨慎地使用该 Utility，否则有可能导致数据或 DB2 子系统的异常。

一般来说，REPAIR Utility 有以下用途：

- TEST 和 REPAIR DBDs；
- RESET TABLESPACE 和 INDEX 的某些 PENDING 状态；

- 验证 TABLESPACE 和 INDEX 的数据区域的内容；
- 替换 TABLESPACE 和 INDEX 的数据区域的内容；
- 删除 TABLESPACE 的某一行记录；
- 为 TABLESPACE 和 INDEX 的数据区域生成 16 位的 DUMP 信息。

4.2.9.2 控制语句及语法

【语法】请参考《DB2 UDB for z/OS V8 Utility Guide and Reference》。

【主要控制语句描述】

1. REPAIR 语句

- **LOG：**指定是否将 REPAIR 带来的变化记入 LOG，默认值为 YES。
- **LEVELID：**当 DATASET 出现 DOWN-LEVELID 的错误时，为 TABLESPACE 或 INDEX 设置一个新的 LEVEL ID。
- **TABLESPACE database-name.table-space-name：**指定设置 LEVELID 的 TABLESPACE 的名称。
- **INDEX：**指定设置 LEVELID 的 INDEX 的名称。
- **PART：**指定设置 LEVELID 的 TABLESPACE 或 INDEX 的 PARTITION 的序号。

2. SET TABLESPACE AND INDEX 语句

- **NOCOPYPEND**：重置 TABLESPACE 或 INDEX 的 COPY-PENDING 状态
- **NORCVRPEND**：重置 TABLESPACE 或 INDEX 的 RECOVER-PENDING 状态
- **NORBDPEND**：重置 TABLESPACE 或 INDEX 的 REBUILD-PENDING 状态
- **NOCHECKPEND**：重置 TABLESPACE 或 INDEX 的 CHECK-PENDING 状态

3. LOCATE 语句

LOCATE BLOCK 和一系列控制语句的组合，它们以 LOCATE 语句开头，以下一个 LOCATE 语句或 SET 语句结尾。一个作业中可以有若干个 LOCATE BLOCK。在一个 LOCATE BLOCK 中可以有若干条 VERIFY，REPLACE 或 DUMP 语句，但只能有一条 DELETE 语句。

LOCATE 语句用来定位 TABLESPACE 中将要被修复的数据。

注意：当 REPAIR 语句后面跟 LOCATE 语句时，在 LOCATE 语句前的操作都被 COMMIT。

4. VERIFY 语句

该语句用来测试某个数据区域中是否包含某个特定的数值。如果存在，则该 LOCATE BLOCK 中后面的语句就允许被执行。如果不存在，则后续的语句被禁止执行。

5. REPLACE 语句

REPLACE 语句将指定位置的数据替换掉。

- **OFFSET：**指定被替换数据的位置。
- **DATA：**定义新的数据，可以用 2～32 位的 16 进制数和字符串表示。

6. DUMP 语句

DUMP 语句用来根据指定的偏移量和长度生成 16 进制的 DUMP。DUMP 语句对 VERIFY 和 REPLACE 语句没有任何影响（无论 VERIFY 成功与否都会执行）。

7. DELETE 语句

DELETE 语句用来删除一条根据 RID 或 KEY 语句定位的行。DELETE 语句必须包含在一个 LOCATE BLOCK 中，常常用来删除 DUPLICATE 的 ROW。

4.2.9.3　参考作业流

1. REPAIR LEVELID

```
REPAIR  LEVELID TABLESPACE  NASEDB.NSPKFKL
```

2. VERIFY 数据并用正确的数据进行覆盖

```
REPAIR OBJECT
LOCATE TABLESPACE TEMPDB.TEMPTS PAGE X'02'
DUMP OFFSET X'0020' LENGTH 20
VERIFY OFFSET X'0020' DATA X'C1D6'
REPLACE OFFSET X'0020' DATA X'D3C9'
DUMP OFFSET X'0020' LENGTH 20
```

3. 删除错误的行

```
LOCATE TABLESPACE TEMPDB.TEMPTS RID(X'0021')
DELETE
```

4.2.10　RUNSTATS Utility

4.2.10.1　Utility功能简介

RUNSTATS Utility 用来收集 TABLESPACE，INDEX，PARTITION，TABLE 的相关信

息，并存储在 DB2 CATALOG 表中。RUNSTATS 有两种形式：RUNSTATS TABLESPACE 和 RUNSTATS INDEX。RUNSTATS TABLESPACE 可以获取包括 TABLESPACE，INDEX，COLUMNS 的信息；而 RUNSTATS INDEX 只获取 INDEX 的相关信息。在 REORG，LOAD，REBUILD INDEX 等作业中使用 STATISTICS 选项，可以进行 INLINE 的 RUNSTATS。

使用 HISTORY 选项，可以收集相关的历史信息。当 DB2 将新的信息存储在 DB2 CATALOG 表中时，同时将历史信息存储在对应一层 HISTORY CATALOG 表中。当对某个对象运行 RUNSTATS 时，DB2 会将 DYNAMIC STATEMENT CACHE 中访问该对象的动态 SQL 语句置为无效。在下次调用该 SQL 语句时将重新选择访问路径。

【运行阶段】如表 4.9 所示。

表 4.9　运行阶段及描述

阶　段	描　述
UTILINIT	初始化 Utility
RUNSTATS	扫描 TABLESPACE 和 INDEX，并更新 CATALOG
UTILTERM	结束

4.2.10.2　控制语句及语法

【语法】请参考《DB2 UDB for z/OS V8 Utility Guide and Reference》。

【主要控制语句描述】

- **TABLESPACE(LIST LISTDEF-NAME)：**指定 TABLESPACE 的名称或包含 TABLESPACE 名的 LIST。
- **PART integer：**指定 RUNSTATS 的 PARTITION。
- **TABLE：**指定哪些 TABLE 的字段信息被收集。这些表必须在上述指定的 TABLESPACE 中。

 ALL：默认值。TABLESPACE 中的所有表；

 TABLENAME：具体指定 TABLE 的名称；

 COLUMN：具体指定 COLUMN 的名称。只有在具体指定了某个 TABLE 的情况下才可以使用该选项。
- **INDEX：**指定获取哪些 INDEX 的信息。通常 INDEX 的第一个字段的信息被收集。其他的字段信息是否在获取之列要看控制语句中的选项。

 ALL：表示获取所有相关 INDEX 的 COLUMN 信息；

INDEXNAME：指定获取某几个 INDEX 的 COLUMN 信息；

PART integer：指定 INDEX 的 PARTITION。

- **SHRLEVEL**：RUNSTATS 过程中，其他程序对 TABLESPACE 和 IDNEX 的访问方式。

 REFERENCE：默认值，允许只读访问；

 CHANGE：允许对 TABLESPACE 和 INDEX 的修改。这种情况下 UNCOMMITED 的数据也会被统计。

- **REPORT**：作业输出一系列关于统计数据的信息。

 NO：默认值，不输出信息；

 YES：输出到 SYSPRINT，通常是 SYSOUT。

- **UPDATE**：是否将统计信息更新到 CATALOG 表中。

 ALL：所有统计数据被更新到 CATALOG 表；

 ACCESSPATH：只将影响访问路径的数据更新到 CATALOG 表；

 SPACE：表示只有有助于 DB2 管理员访问特定的 TABLESPACE 或 INDEX 的 COLUMN 被更新；

 NONE：所有数据都不更新。该选项只有在 REPORT YES 时才有效。

- **KEYCARD**：统计 INDEX 中所有 COLUMN 的 DISTINCT 值。

4.2.10.3　如何运行RUNSTATS Utility

1. 什么时候使用 RUNSTATS

- 在表 LOAD 完数据后。
- 创建 INDEX 后。
- 对 TABLESPACE 做完 REORG 而没有进行 INLINE 的 STATISTICS 的统计时。
- 在对 TABLESPACE 做完大量的 UPDATE，INSERT，DELETE 等操作后。
- 在做完 RECOVER TABLESPACE，REBUILD INDEX 和 REORG INDEX 而没有进行 INLINE 的 STATISTICS 的统计时。
- 在带 OFFPOSLIMIT，INDREFLIMIT，LEAFDISTLIMIT 等选项进行 REORG 前。

也可以对 DB2 的 CATALOG TABLESPACE 做 RUNSTATS。

2. 访问路径相关的数据

在“用途”栏中为“S”的表示该字段信息将用于 DB2 优化器分析访问路径，“G”表示提供给用户的参考信息。

- SYSIBM.SYSTABLES

字　段	描　述	用　途
CARDF	表中的总行数	S
NPAGES	表中的数据所占用的 PAGE 总数	S
NPAGESF	表使用的 PAGE 总数	S
PCTROWCOMP	被压缩的行的百分比	S
STATSTIME	最后一次 RUNSTATS 更新的时间	G

- SYSIBM.SYSTABSTATS

字　段	描　述	用　途
CARD 或 CARDF	该 PARTITION 中的总行数	S
NPAGES	PARTITION 中数据所占用的 PAGE 总数	S

- SYSIBM.SYSCOLUMNS

字　段	描　述	用　途
COLCARDF	该字段的 DISTINCT 值的总数的估计值。该值为-1 表示数据尚未被统计； -2 表示该列是在一个 AUXILIARY 表中	S
HIGH2KEY	该列的第二高的值。该值为空表示数据尚未统计，或该列在一个 AUXILIARY 表中。该列可以被修改	S
LOW2KEY	该列的第二低的值。该值为空表示数据尚未统计，或该列在一个 AUXILIARY 表中。该列可以被修改	S
STATSTIME	该表最后一次被 RUNSTATS 更新的时间	G

- SYSIBM.SYSCOLDIST

字　段	描　述	用　途
CARDF	COLUMN GROUP 的 DISTINCT 值的总数	S
COLGROUPCOLNO	与该列统计数据相关的 COLUMN 集合	S
COLVALUE	根据分布的 INDEX 数据计算出来的列的实际值	S
FREQUENCYF	含有 COLVALUE 的值的行的百分比	S
NUMCOLUMNS	与 COLUMN 统计数据相关的 COLUNMN 数量	G
STATSTIME	该表最后一次被 RUNSTATS 更新的时间	G

- SYSIBM.SYSTABLESPACE

字　段	描　述	用　途
NACTIVE 或 NACTIVEF	TABLESPACE 中 ACTIVE PAGE 的总数。值为 -1 表示数据未作统计	S
STATSTIME	该表最后一次被 RUNSTATS 更新的时间	G
DSSIZE	文件大小（KB）	G

- SYSIBM.SYSINDEXE

字　段	描　述	用　途
CLUSTERRATIOF	按 CLUSTER 顺序存放的数据的百分比，取值范围 0 ～ 1	S
CLUSTERING	是否为 CLUSTERING INDEX	S
FIRSTKEYCARDF	INDEX 第一个字段的唯一值的统计总数	S
FULLKEYCARDF	FULL KEYS 的唯一值的统计总数	S
NLEAF	LEAF PAGE 的总数	S
NLEVELS	INDEX TREE 的层数	S
STATSTIME	该表最后一次被 RUNSTATS 更新的时间	G

- SYSIBM.SYSINDEXSTATS

字　段	描　述	用　途
CLUSTERRATIOF	按 CLUSTER 顺序存放的数据的百分比，取值范围 0 ～ 1	S
FIRSTKEYCARD 或 FIRSTKEYCARDF	该 INDEX PARTITION 中第一个字段的唯一值的统计总数	S
FULLKEYCARD 或 FULLKEYCARDF	该 INDEX PARTITION 中 FULL KEYS 的唯一值的统计总数	S
NLEAF	该 INDEX PARTITION 中 LEAF PAGE 的总数	S
NLEVELS	该 INDEX PARTITION 中 INDEX TREE 的层数	S
KEYCOUNT	PARTITION 中的行数	S

4.2.10.4　参考作业流

1. 更新 TABLESPACE 的 CATALOG 数据并允许读写访问

```
//PH01S26 EXEC DSNUPROC,PARM='SB0A,STAT',COND=(4,LT)
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR
//SORTOUT  DD  UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
```

```
//SORTWK01 DD  UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SORTWK02 DD  UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SORTWK03 DD  UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//DSNTRACE DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(4000,(50,50),,,ROUND)
//SYSREC   DD  UNIT=SYSDA,SPACE=(4000,(200,200),,,ROUND)
//SYSIN    DD  *
 RUNSTATS TABLESPACE PRASDB.PSFXMAN TABLE(ALL) SAMPLE 25
 INDEX(ALL) SHRLEVEL CHANGE
```

2. 对 INDEX 做 RUNSTATS

```
RUNSTATS INDEX (NGDBA.PIHFXMAN1)
```

3. 更新 TABLESPACE 中的所有统计信息（TABLESPAE,TABLE,INDEX, COLUMN）

```
RUNSTATS TABLESPACE (NASEDB.NSPATEL) TABLE INDEX
```

4. 更新与访问路径相关的统计信息

```
RUNSTATS TABLESPACE PRASDB.PSFXMAN
REPORT YES
UPDATE ACCESSPATH
```

5. 更新一个 PARTITION 的 CATALOG 数据

```
RUNSTATS TABLESPACE PRASDB.PSFXMAN PART 1
INDEX (NGDBA.PIHFXMAN1 PART 1)
REPORT YES
```

4.2.11 TEMPLATE Utility

4.2.11.1 Utility功能简介

TEMPLATE Utility 控制语句可以在使用 LISTDEF 时不通过 JCL 语句来分配文件。在控制语句中可以包含分配文件用到的参数：文件大小、所在位置、文件属性等。

【输出】动态生成一个的指定名称的 TEMPLATE，可以在以后引用。

【运行阶段】

只有一个 UTILINIT 的阶段。

4.2.11.2 控制语句及语法

【语法】请参考《DB2 UDB for z/OS V8 Utility Guide and Reference》。

【主要控制语句描述】

- **TEMPLATE template-name**：定义 TEMPLATE 的名称。
- **UNIT**：指定文件所在的设备名称，可选值为 DASD 和 TYPE，默认值为 SYSALLDA。
- **DSN name-expression**：指定 TEMPLATE 中包含的文件的名称。可以使用变量，数字和字母。其中变量用(&)开头，不同变量之间用（..）分隔。例如：DSN &DB..&TS..D&JDATE..COPY&ICTYPE。 TEMPLATE Utility 还提供了十分丰富的变量，具体请参考《DB2 UDB for z/OS V8 Utility Guide and Reference》。
- **DISP**：文件的 DISPOSITION 参数，和 JCL 中的设置一样。
- **RETPD interger**：指定文件保留的时间，范围为 1～9 999。如果指定了 DATACLAS、MGMTCLAS、STORCLAS，可以省略此参数。
- VOLUMES (vol1,vol2,...)：指定文件存放的卷的名称。
- **SPACE (primary,secondary)**：指定文件的首次分配空间和二次分配的空间的大小，用法和 JCL 中的一样。
- **STACK**：指定输出的文件是否连续地分配在相同的卷上。

 NO：默认值，输出文件不在相同的卷上。

 YES：将一系列类似的输出文件，例如 IMAGE COPY 文件写到一系列相同类型的卷上。

4.2.11.3　使用TEMPLATE需注意的几个问题

1. TEMPLATE 中文件的命名规则

TEMPLATE 中创建的文件必须是唯一的、有实际意义的。参考以下的规则：

- 通常使用数字，字母和一些变量名来组成数据集的 QUALIFY；
- 使用双字节的变量名来节省空间；
- 如果变量名处于一个 QUALIFY 的结尾处，则在变量名后使用两个“..”，如：&DB..&TS.；
- 使用变量&DB.，&TS.使文件和数据库的 OBJECT 相对应；
- 使用 P&PART 表示 TABLESPACE 的 PART；
- 使用&JO.和&ST.表示作业和作业步；
- 如果需要在文件名中使用到子系统、MEMBER、用户、Utility-ID 以及 Utility NAME，分别使用&SS.&US.和&UT.等变量来代替。

- 使用 D&DATE.和 T&TIME.等变量来保证文件名的唯一性。
- 使用&IC.、&LR.和&PB.等变量来定义 IMAGE COPY 文件的名称。

2. 使用磁带

通过 STACK 参数，我们可以使磁带以如下两种方式工作。

（1）STACK = NO 时，磁带以传统的单文件的方式工作，文件被写到磁带上，磁带被 REWIND，REPOSITION 和 REMOUNT。

（2）STACK = YES，文件被连续的写到一个单独的磁带上，不需要 REPOSITION 和 REMOUNT 磁带。使用这种方法，需要考虑下面的因素：

- 只有“类似的文件”能够被 STACK 到一个磁带上。例如，一个磁带可以存放一系列本地的 IMAGE COPY 文件，另一个磁带可以存放远程的 IMAGE COPY 文件。
- 必须是在同一个 Utility 调用中，当 Utility 中止时，STACK 也会被中断掉。对于多个 OBJECT，如果希望它们分配到同一个磁带上，可以使用 LISTDEF 使它们能在同一个 Utility 中运行。
- 最好不要在有并行操作的 Utility 中使用 STACK YES，否则会造成冲突。
- 当一个磁带写满后，STACK 会被中止，这时会有第二个卷被 MOUNT。

4.2.11.4 参考作业流

1. 使用 TEMPLATE 做 IMAGECOPY 到硬盘上

```
TEMPLATE COPYDS DSN
 &DB..&TS..COPY&IC.&LR.&PB..D&DATE..T&TIME.
```

2. 在磁带上指定一个 TEMPLATE 名，并限制保留期限

```
TEMPLATE TAPEDS DSN(&DB..&TS..D&DATE.)
     UNIT 3390  RETPD 30
```

3. 在硬盘上指定一个 TEMPLATE 名，并指定分配空间

```
TEMPLATE DISK DSN &DB..&TS..T&TIME.
     SPACE(100,10) CYL
```

4. 用 TEMPLATE 创建 GDG 文件，并在 IMAGE COPY 中使用 GDG 文件

```
LISTDEF CPY1 INCLUDE TABLESPACE ****DB.****TS
           INCLUDE TABLESPACE ****DB.****TS
TEMPLATE TMPT UNIT VTAPE RETPD 30
       DSN (BACKUP.PB01.&DB..&TS..D&DT.)
```

```
        DISP (NEW,CATLG,DELETE) STACK YES
TEMPLATE TMPD
        DSN 'AYX.PB01.&DB..&TS.(+1)'
        UNIT 3390
        DISP (NEW,CATLG,DELETE) GDGLIMIT(1)
        MAXPRIME 1000
COPY LIST CPY1 COPYDDN (TMPT,TMPD) PARALLEL (5) TAPEUNITS(5)
SHRLEVEL CHANGE
```

4.2.12　UNLOAD Utility

4.2.12.1　Utility功能简介

UNLOAD 工具比 DB2 自带的程序 DSNTIAUL 效率更高，功能更多。它可以从若干个数据源 UNLOAD 数据到若干个 BSAM sequential data sets 中。数据源可以是 DB2 TABLESPACE 或者 DB2 IMAGE COPY DATASET。

使用 UNLOAD，可以从整个 TABLESPACE 中 UNLOAD 记录或者从指定的 PARTITION、TABLE 上进行 UNLOAD。如果一个 TABLESPACE 是分 PARTITION 的，可以 UNLOAD 所有选定的 PARTITION 到一个 DATASET 中，或者并发的同时 UNLOAD 所有的 PARTITION 到物理上分开的若干个 DATASET 中。

【运行阶段】如表 4.10 所示。

表 4.10　运行阶段及描述

阶　段	描　述
UTILINIT	初始化 Utility
UNLOAD	将数据 UNLOAD 到顺序文件。当 UNLOAD 在对 TABLESPACE 或 PARTITION 进行操作时，DB2 会进行 INTERNAL 的 COMMIT。这样如果在这个阶段作业中断后重启，就可以获得 COMMIT 点
UTILTERM	清除 Utility

4.2.12.2　控制语句及语法

【语法】请参考《DB2 UDB for z/OS V8 Utility Guide and Reference》。

【主要控制语句描述】

- **DATA：**表示从 TABLE 中 UNLOAD 数据，与 TABLESPACE、PART、LIST 等关键字互斥。当使用了 DATA 关键字，必须和 FROM TABLE 关键字连用。例如：UNLOAD

DATA FROM TABLE T1.PERSON。

- **TABLESPACE**：指定从某个 TABLESPACE 中 UNLOAD 数据，该 TABLESPACE 不能是 LOB TABLESPACE。
- **PART**：指定从某个或某几个 PARTITION 上 UNLOAD 数据。

 integer：指定 PARTITION 的序号。

 Int1：**Int2** 指定一个 PARTITION 的区间，例如 2:4 表示 2，3，4 三个 PARTITION。Int1 必须小于 Int2。

如果没有指定 PART 参数，则从整个 TABLESPACE 中 UNLOAD 数据。

- **FROMCOPY data-set-name**：指定从某个 IMAGE COPY 文件中 UNLOAD 数据。使用 FROMCOPY 只对一个 IMAGE COPY 文件进行操作；而使用 FROMCOPYDDN 则可对多个文件 IMAGE COPY 文件进行操作。
- **FROMCOPY ddname**：表示从若干 IMAGE COOY 文件中 UNLOAD 数据。这些 IMAGE COPY 文件可以用一个 DDNAME 连接起来。
- **LIST listdef-name**：指定用 LISTDEF 定义的 LIST 名。当使用 LIST 时，需要使用 TEMPLATE 定义 PUNCH 文件。
- **PUNCHDDN**：指定存放 UNLOAD Utility 生成的 LOAD 控制语句的文件，或者某个定义了一组文件的 TEMPLATE 的名称。

 ddname：指定文件名，默认为 SYSPUNCH。如果用户对多个 TABLESPACE 或 TABLE 做 UNLOAD，而且希望生成 LOAD 控制语句，就需要定义一个合适的 TEMPLATE 来对应多个 PUNCH 文件。如果 PUNCHDDN 参数省略，则无法得到 LOAD 控制语句。
- **UNLDDN**：指定存放 UNLOAD 数据的文件名或 TEMPLATE 的名称。

 ddname：指定 DD NAME，默认为 SYSREC。

 template-name：指定 TEMPLATE 的名字。如果一个指定的名字在 JCL 中是一个 DD NAME，而在 TEMPLATE 中被定义成 TEMPLATE，则它被当成 DD NAME 使用。
- **SHRLEVEL**：指定当 Utility 运行时其他的应用进程可否访问该 TABLESPACE 或 PARTITION。当数据源为 IMAGE COPY 文件时，此参数被忽略。默认值为 **SHRLEVEL CHANGE ISOLATION CS**。

 CHANGE：当 UNLOAD 运行时表中的数据可以被读、插入、修改、删除。

 ISOLATION：指定隔离级别。可选值为 CS 和 UR。

REFERENCE：当 UNLOAD 运行时表中的数据为只读，不能进行插入、修改、删除等操作。

- **FROM TABLE 语句**：

一个 TABLESPACE 的所有表和 PARTITION 的数据都可以在一个单独的 UNLOAD UTITLIY 中 UNLOAD 出来。在一条 FROM TABLE 语句中，所有的表必须在同一个 TABLESPACE 中。如果需要 UNLOAD 某个特定字段，就必须在 FROM TABLE 语句中指定每个表的名字。如果对于一个 TABLESPACE 没有指定 FROM TABLE 语句，则 TABLESPACE 中所有的行都会被 UNLOAD 出来。

注意：如果前面指定了 LIST 选项，则不能使用 FROM TABLE 语句。

Table-name：指定从这个 TABLE 中 UNLOAD 数据。如果没有指定 QUALIFY，则用户名作为 QUALIFY。

HEADER：在每条输出记录前加上 HEADER 指定的字符串，同时会使 SYSPUNCH 文件中加上 WHEN 语句，例如 WHEN(00001:00003 = 'ABC')。默认为 HEAD OBID。

4.2.12.3　如何运行UNLOAD

Utility 中使用的 DATA SET 如表 4.11 所示。

表 4.11　Utility 中使用的 DATA SET

DATA SET	描　　述	是否必需
SYSIN	存放控制语句的文件	是
SYSPRINT	输出信息的文件	是
INPUT DATA SET	用来做 UNLOAD 的数据源文件	是
SYSPUNCH	若干存放输出的 LOAD 控制语句的文件	是
UNLOADED DATA SET	若干存放 UNLOAD 数据的工作文件	是

4.2.12.4　参考作业流

1. 从 TABLESPACE 中 UNLOAD 所有表的所有字段

```
//UNLOAD1 JOB (ACCTNUM,EXP),'PGMRNAME',
//        TIME=1440,
//        NOTIFY=&SYSUID,
//        REGION=0M,
//        CLASS=B,
```

```
//        MSGCLASS=X,
//        MSGLEVEL=(1,1)
//STEP1 EXEC DSNUPROC,SYSTEM=DB1E,UID='UNLOAD1',UTPROC=''
//SYSREC DD DSN=ZHS0013.UNLOAD.TEST1,
//    DISP=SHR
//SYSPUNCH DD DSN=ZHS0013.DB2.JCL(UNLOAD1),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD  *
   UNLOAD TABLESPACE TESTDB.TESTTS4
```

2. 从 TABLESPACE 中的某个表 UNLOAD 数据

```
//UNLOAD1 JOB (ACCTNUM,EXP),'PGMRNAME',
//        TIME=1440,
//        NOTIFY=&SYSUID,
//        REGION=0M,
//        CLASS=B,
//        MSGCLASS=X,
//        MSGLEVEL=(1,1)
//STEP1 EXEC DSNUPROC,SYSTEM=DB1E,UID='UNLOAD1',UTPROC=''
//SYSREC DD DSN=ZHS0013.UNLOAD.TEST1,
//    DISP=SHR
//SYSPUNCH DD DSN=ZHS0013.DB2.JCL(UNLOAD1),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD  *
   UNLOAD DATA FROM TABLE T1.PERSON3
```

3. 从某个表中 UNLOAD 指定的字段，控制语句如下：

```
UNLOAD TABLESPACE DSN8D71A.DSN8S71E NOPAD
  FROM TABLE DSN8710.EMP
    (EMPNO, LASTNAME, SALARY DECIMAL EXTERNAL)
  WHEN (WORKDEPT = 'D11' AND SALARY > 25000)
```

4. 使用 LISTDEF 有选择的 UNLOAD TABLESPACE 的若干 PARTITION

```
//SAMPJOB  JOB ...
//STEP1    EXEC DSNUPROC,UID='SMPLUNLD',UTPROC='',
//              SYSTEM='V71A'
//SYSPUNCH DD DSN=USERID.SMPLUNLD.SYSPUNCH,
//        DISP=(NEW,CATLG,CATLG),
//        UNIT=SYSDA,SPACE=(TRK,(1,1))
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  LISTDEF  UNLDLIST
```

```
        INCLUDE  TABLESPACE TDB1.TSP1 PARTLEVEL(1)
        INCLUDE  TABLESPACE TDB1.TSP1 PARTLEVEL(3)
  TEMPLATE UNLDDS   DSN &USERID..SMPLUNLD.&TS..P&PART.
        UNIT SYSDA  DISP (NEW,CATLG,CATLG) SPACE (2,1) CYL
  UNLOAD LIST UNLDLIST
    UNLDDN UNLDDS
```

4.2.13　监控与控制DB2 Online Utility

4.2.13.1　监控命令及Utility状态控制

可以使用以下方法检测和控制 DB2 Utility。

- DISPLAY 命令：可以监控 Utility 当前的运行情况和状态。
- TERMINATE 命令：可以终止某个正在执行或已经异常中断的 Utility。

DB2 Utility 的状态有以下几种。

- 激活(Active)：Utility 正在处理中。
- 暂停(Stop)：工具被暂停，被改变过的数据不可用。
- 终止(Terminating)：工具被 TERMINATE Utility 命令终止。

一个处于暂停状态的工具可以用 RESTART 参数重启。RESTART 可以从两点重启（自 DB2 V8 版本以后，DB2 会自动与 Current 点重新开始暂停的 Utility，无须指定 Restart 参数）。

- PHASE：从执行的最后一个状态（PHASE）重新开始。
- CURRENT：在最后的内部提交点（COMMIT POINT）重新开始。提交点的信息保存在 SYSUTILX 表空间中。

4.2.13.2　SYSUTILX 表空间

SYSUTILX 表空间包含两个表：

- SYSUTIL
- SYSUTILX

SYSUTIL 记录的作用如下：

- 阻止其他 Utility 同时处理相同的 DB2 对象。
- 在 Utility 的提交点（COMMIT POINT）或断点（STOP POINT）被更新，这样在 Utility 重新开始的时候就能得到启动的信息。

SYSUTILX 表在 CHECKPOINT/RESTART 信息量超过 SYSUTIL 表的可用空间时保存溢出信息。

TERMINATE Utility 命令用于从 SYSUTILX 目录表空间中删除特定 UtilityID 的行。对大多数用这种方式终止的 Utility，其 DB2 对象是可以访问的。某些 Utility，如 COPY，LOAD，RECOVER 和 REORG，在 DB2 对象可用之前需要执行一些必要的步骤。

SYSUTILX 是 DB2 DIRECTORY 表，用户不能用 SQL 语句查询。要查询表信息，用户需使用 DISPLAY Utility 命令；删除行使用 TERMINATE Utility 命令。



4.3 DB2 Stand Alone Utility

4.3.1 如何调用Stand Alone Utility

1．创建作业流

下列 Stand Alone Utility 从 SYSIN 中读取控制语句：

Utility	DDNAME
DSNJU003	SYSIN
DSNJU004	SYSIN
DSN1LOGP	SYSIN
DSN1SDMP	SDMPIN

2．控制语句书写规则

- 输入的控制语句为 LRECL=80 格式，73 列后的语句被忽略。
- 控制语句在解析前被拼接成一条单独的语句，换行不需要使用连接符。
- SYSIN 中可以有多条 Utility 控制语句。

3．下列Utility可以使用EXEC PARM

```
DSN1CHKR, DSN1COMP, DSN1COPY, DSN1PRNT
```

4.3.2 DSN1COPY Utility

4.3.2.1 Utility功能简介

DSN1COPY 的用途有：

- 将 DB2 VSAM 文件复制到一个 PS 文件；

- 将一个 PS 文件复制到 DB2 VSAM 文件；
- 将 DB2 IMAGE COPY 文件复制到 DB2 VSAM 文件；
- 在 DB2 VSAM 文件之间互相复制；
- 将一个 PS 文件复制到另一个 PS 文件。

4.3.2.2　控制语句和语法

【语法】请参考《DB2 UDB for z/OS V8 Utility Guide and Reference》。

【主要控制语句描述】

- **CHECK：**检查 SYSUT1 指定文件的中每个 PAGE 的合法性。如果发现错误，则 DSN1COPY 报出错误类型并将 PAGE 的 16 进制数的 DUMP 信息输出到 SYSPRINT。如果某个 PAGE 上有不止一个错误，只会报第一个错误。
- **32K：**表示输入文件为 32KB PAGE。推荐使用 PAGESIZE(32KB)。
- **PAGESIZE：**指定输入的文件的 PAGE 大小。如果指定得不对，DSN1COMP 的结果将不可预知。PAGESIZE 这个参数 DB2 支持的取值为 4K，8K，16K，32K。如果没有指定该参数，DSN1COPY 会尝试确定输入文件的 PAGE 大小；如果 DSN1COPY 不能确定，则会报错误信息。
- **FULLCOPY：**表示输入文件为 DB2 FULL IMAGE COPY 文件。如果输入文件为 IMAGECOPY 文件而省略该参数，则 DB2 有可能会报错。如果输入文件为 IMAGECOPY 文件而没有指定该参数，则 DSN1COPY 有可能会报错。使用 FULLCOPY 要求作业流中 SYSUT2 文件必须是 DB2 VSAM 文件或 DUMMY。
- **SEGMENT：**表示输入文件为 SEGMENT TABLESPACE。不能和 FULLCOPY、INCRCOPY 参数一起使用。
- **LARGE：**表示输入的 TABLESPACE 定义时使用了 LARGE 关键字，DB2 认为该文件大小为 4GB。推荐使用 DSSIZE(4GB)表示该文件是 LARGE TABLESPACE。
- **NUMPARTS(integer)：**指定输入文件的 PARTITION 的总数，可选的值为 1～254。DSN1COPY 使用该值来计算输出文件的大小及将要打印的第一个 PAGE。如果省略该参数或设为 0，则 DSN1COPY 认为该 TABLESPACE 为非 PARTITION 的。如果参数指定错误，则 DSN1COPY 可能会操作错误的文件。
- **PIECESIZ(integer)：**指定 NON-PARTITIONED INDEX 文件的 PIECE 大小。非 LARGE TABLESPACE 的 INDEX 的默认值为 2GB，LARGE TABLESPACE 的 INDEX

默认值为 4GB。PIECESIZ 的取值单位有 K（表示 **integer** 的值是乘以 1KB 的结果，**integer** 必须是 256 或 512），M（表示 **integer** 的值是乘以 1MB 的结果，**integer** 必须是 1 到 512 之间的 2 的倍数），G（表示 **integer** 的值是乘以 1GB 的结果，**integer** 必须是 1、2 或 4）。有效的 PIECESIZ 值有：1MB 或 1GB；2MB 或 2GB；4MB 或 4GB；8MB；16MB；32MB；64MB；128MB；256KB 或 256MB；512KB 或 512MB。

- **OBIDXLAT**：DB2 文件在复制之前将转换其 OBID。必须在 SYSXLAT 文件中输入相关的转换信息如 DBID、PSID、OBID 等。DSN1COPY 可以同时转换 500 个 OBID。当指定了 OBIDXLAT 选项后，DSN1COPY 将执行 CHECK，无论 CHECK 参数是否指定。
- **RESET**：将 HEAD PAGE 中的 HIGH FORMAT 的 PAGE NUMBER 以及 LOG RBA 设为 0。当使用了该参数后，DSN1COPY 将执行 CHECK，无论 CHECK 参数是否指定。当你的 DB2 SUBSYSTEM 中的 TABLESPACE 文件用另一个 DB2 SUBSYSTEM 的 DB2 文件的 DSN1COPY 输出来构成，必须使用该参数。否则会产生文件 DOWN-LEVELID 的错误。

4.3.2.3 如何运行DSN1COPY Utility

1. DSN1COPY 的运行环境

DSN1COPY Utility 以 MVS 批量作业的方式运行。可以在 DB2 子系统运行或停止状态时运行。

当 DB2 处于正在运行状态时，按照下列步骤执行：

- 将 TABLESPACE 置为只读状态，START DB(XXXX) SP(XXXX) ACCESS(RO)。
- 用 WRITE(YES)选项对 TABLESPACE 做 QUIESCE。
- 运行 DSDN1COPY，进行文件级别的复制。
- 将 TABLESPACE 启动到正常状态，START DB(XXXX) SP(XXXX) ACCESS(RW)。

2. 作业流中需要的文件

- **SYSPRINT**：输出文件，包含所有的输出信息及 16 进制的 DUMP 信息。
- **SYSUT1**：输入文件，可以是用 DSN1COPY 生成的 PS 文件，或者是 VSAM 文件。DSN1COPY 认为文件的 BLOCK SIZE 为 4096 个字节。除非 DB2 DATABASE 被 STOP，最好是用 DISP=OLD 以保证该文件没有被 DB2 使用。如果输入文件为 PARTITION 的 TABLESPACE 或 IDNEX，则需要指定该 TABLESPACE 的 NUMPARTS。如输入文件是一个 4 个 PARTITION 的 TABLESPACE 的第二个

PARTITION，需要指定 NUMPARTS(4)，并且输入文件名如 DSN=XXXX……A002 的格式。

- **SYSUT2**：定义输出文件。可以是 PS，VSAM 文件，也可以是 DUMMY。DSN1COPY 认为 SYSUT2 文件是空文件。当对输入文件做 DUMP 或 CHECK 时，SYSUT2 可以是 DUMMY。DSN1COPY 通过 ICF CATALOG 可以获取必要的信息，不需要指定 UNIT 和 VOLUME 参数。
- **SYSXLAT**：用来进行 DBID，OBID，PSID 以及 ISOBID 的转换。如果在一个 TABLESPACE 中删除了一个 TABLE，而没有做 REORG，那么在运行 DSN1COPY 前要先运行 REORG 将 TABLESPACE 中多余的记录删掉。在 SYSXLAT 文件中，每条记录有一对数字，前者为源对象对应值，后者为目标对象对应值。第一条记录为 DBID，取值范围为-32 767～65 535；第二条记录为 PSID/ISOBID，取值范围为 0～32 767；其后所有的记录都是 OBID。

3. 使用 DSN1COPY 的一些注意事项

- **ALTER A TABLE BEFORE DSN1COPY**：当修改一个表结构，比如 ALTER TABLE ADD COLUMN 后，必须运行 REORG 作业才能使用 DSN1COPY。
- **TRANSLATE DB2 INTERNAL IDENTIFIER**：使用 DSN1COPY 而不指定 OBIDXLAT 可能会导致 OBID 不可用。例如在下列的情况下：
 - — 当用户用 DSN1COPY 创建数据后，在使用数据之前对表进行删除和重建。
 - — 当源 DB2 子系统和目标 DB2 子系统有下列差异：
 - ➢ TABLESPACE 的 BUFFERPOOL 和 NUMPARTS 参数。
 - ➢ TABLE 中有除了 NAME，TABLESPACE NAME 和 DATABASE NAME 的参数。
 - ➢ 用户在源 DB2 子系统和目标 DB2 子系统中定义和删除这些 TABLESPACE，INDEX，TABLE 的顺序。
- **COPY MULTIPLE DATASET TABLESPACE**：当使用 DSN1COPY 将 IMAGE COPY 文件恢复到一个 DB2 文件时，按如下方式指定 SYSUT2 文件：
 - — 如果 SYSUT1 为单个 PARTITION 的 IMAGECOPY，SYSUT2 必须为 TABLESPACE 的该 PARTITION 的文件名。并指定 NUMPARTS(nn)参数，nn 为 TABLESPACE 的 PARTITION 的总数。
 - — 如果 SYSUT1 为 TABLESPACE 所有 PARTITION 的 IMAGECOPY，SYSUT2 必须为 TABLESPACE 的第一个 PARTITION 的文件名，并指定 NUMPARTS(nn)参

数，nn 为 TABLESPACE 的 PARTITION 的总数。

— 如果 SYSUT1 为一个有多文件的 LINEAR TABLESPACE 的某个文件的 IMAGECOPY，SYSUT2 必须为那个输出的文件名，不需要指定 NUMPARTS 参数。

— 如果 SYSUT1 为一个有多文件的 LINEAR TABLESPACE 的所有文件的 IMAGECOPY，SYSUT2 必须为该 TABLESPACE 的第一个文件名。

4.3.2.4 参考作业流

1. 运行 DSN1COPY，使用 CHECK 参数

```
//DSN1COPY JOB (ACCTNUM,EXP),'PGMRNAME',
//        TIME=1440,
//        NOTIFY=&SYSUID,
//        REGION=0M,
//        CLASS=B,
//        MSGCLASS=X,
//        MSGLEVEL=(1,1)
//RUNCOPY EXEC PGM=DSN1COPY,PARM='CHECK'
//STEPLIB  DD DSN=DSN810.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUT1  DD DSN=DSNDB0E.DSNDBC.DSNDB01.SYSUTILX.I0001.A001,
//         DISP=OLD
//SYSUT2   DD DSN=TEMP.DS,UNIT=3390,DISP=(NEW,KEEP),
//         SPACE=(CYL,(1,1))
```

2. 使用 OBIDXLAT 参数进行 OBID 转换

```
//STEP1 EXEC PGM=DSN1COPY,
//     PARM='OBIDXLAT,FULLCOPY,LARGE,RESET,NUMPARTS(224)',
//     REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSUT1 DD DISP=SHR,DSN=ASBK.PB01.ECISDB.TS1170.P00171(0)
//SYSUT2 DD DISP=OLD,
//          DSN=DB2CE01.DSNDBC.ECISDB.TS1170.I0001.A171
//SYSXLAT  DD *
 DBID 286 283
 PSID 68 36
 OBID 69 37
```

4.3.3　DSN1LOGP Utility

4.3.3.1　Utility功能简介

DSN1LOGP 可按照两种格式打印 RECOVERY LOG 的信息：DETAIL REPORT 和 SUMMARY REPORT。SUMMARY REPORT 可以帮助用户进行 CONDITIONAL RESTART，INDOUBT THREAD 的处理和检查 DATA PROPAGATION 的问题。用户可以指定 LOG 的范围，如可以指定 URID 和 DATABASE。

4.3.3.2　控制语句和语法

【语法】请参考《DB2 UDB for z/OS V8 Utility Guide and Reference》。

【主要控制语句描述】

- **RBASTART(hex-constant)**：指定读取 LOG 的起始位置的 RBA 值。如果该值不能与 LOG 中的某个 RBA 值匹配，DSN1LOGP 将从该值后面的 RBA 开始读取。该参数只能出现一次，可以简写为 ST。HEX-CONSTANT 为表示 16 进制数的 12 位的字符串（6 个字节），默认值为 0。
- **RBAEND(hex-constant)**：指定读取的结束的 RBA 的值。默认值为 FFFFFFFFFFFF，这将会读取到 LOG 的结尾。
- LRSNSTART(hex-constant)：指定开始从某个 LRSN 开始扫描 LOG。
- LRSNEND(hex-constant)：指定扫描 LOG 到这个 LRSN 结束。
- **DBID(hex-constant)**：指定一个 DBID，DSN1LOGP 只抽取与该 DBID 相关的 LOG 记录。Hex-constant 为 1～4 位的字符。DBID 的值可以通过查询 DB2 CATALOG 表 SYSIBM.SYSTABLESPACE 得到。
- **OBID(hex-constant)**：指定一个 1～4 位的 OBID 值，DSN1LOGP 将只列出与该 OBID 相关的 LOG 记录。OBID 的值可以通过查询 DB2 CATALOG 表 SYSIBM.SYSTABLESPACE 中的 PSID 得到，需要转换为 16 进制。
- **PAGE(hex-constant)**：指定一个 PAGE 号。只查询相关 PAGE 的 LOG 记录。PAGE 和 RID 不能同时指定。
- **RID(hex-constant)**：指定一个 10 位字符的 RECORD ID。前 8 位字符表示 PAGE NUMBER，后 2 位表示 RID。DSN1LOGP 不仅仅打印与该 RECORD 直接相关的 LOG

记录，如 INSERT 和 DELETE 等操作，也会打印出与指定的 DBID 和 OBID 相关的控制信息，如文件的打开和关闭等操作。在一个作业中至多可以指定 40 条 RID。

- **SUMMARY**：默认值为 SUMMARY(NO)。
 (YES)：同时生成 DETAIL 和 SUMMARY 报告；
 (NO)：只生成 DETAIL 报告；
 (ONLY)：只生成 SUMMARY 报告。
- **FILTER**：使生成的 SUMMARY 报告内容只包含指定的 URID 和 LUWID 相关的 LOG 信息。

4.3.3.3 如何运行DSN1LOGP作业

作业中需要分配的文件如下。

- SYSPRINT：DSN1LOGP 将所有错误信息，EXCEPTION CONDITION， DETAIL REPORT 写到 SYSPRINT 文件，LRECL=131。
- SYSIN：输入控制语句的文件，LRECL=80，控制语句必须在 1～72 列之间。最多支持 50 条控制语句。
- SYSSUMRY：存放输出的 SUMMARY REPORT 信息。

下面是指定 LOG 文件的 DD 语句：

- BSDS：BSDS 文件可以提供系统中所有 ARCHIVE LOG 和 ACTIVE LOG 的信息。在指定了一个 BSDS 文件后，必须提供显示的起始 RBA 和结束 RBA 值。
- ACTIVEn：如果 BSDS 文件不可用，可以用 ACTIVEnn DD 语句指定一个或多个 ACTIVE LOG 文件。如果指定了不止一个 ACTIVE LOG 文件，必须按顺序给出各个 ACTIVE LOG 的 RBA 值范围。例如 ACITVE1 LOG 的 RBA 要小于 ACTIVE2 LOG；ACTIVE2 LOG 的 RBA 要小于 ACTIVE3 LOG。如果使用 ACTIVE LOG，不需要在控制语句中指定 RBASTART 和 RBAEND。
- ARCHIVE：在 ARCHIVE DD 语句下，可以指定多个 ARCHIVE LOG 文件，这些 ARCHIVE LOG 文件必须按照 RBA 值的升序排列，否则会出错。DSN1LOGP 会扫描列出所有的 ARCHIVE LOG 文件。

在 DATASHARING 环境下，使用下面的文件命名：

```
GROUP、MxxBSDS、MxxACTn、MxxARCHV
```

GROUP 对整个 DB2 GROUP 有效，当使用 GROUP 时，必须指定起始和结束的 LRSN，

而不是 RBA 值，否则会出错。Mxx 表示是第几个 MEMBER。

4.3.3.4　参考作业流

1. 打印 BSDS 信息

```
//PRINT2 EXEC PGM=DSN1LOGP
//STEPLIB  DD DSN=DSN810.SDSNLOAD,DISP=SHR
//SYSABEND DD SYSOUT=*
//SYSSUMRY DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//BSDS     DD DSN=DSNPB01.PB11.BSDS01,DISP=SHR
//*ACTIVE1  DD DISP=SHR,DSN=DSNPB01.PB11.LOGCOPY1.DS01
//SYSIN    DD *
 STARTRBA(003681820091) ENDRBA(0036818215D6) SUMMARY(ONLY)
 DBID(103) OBID(4)
```

2. 打印 ACTIVE LOG 信息（没有可用的 BSDS 文件）

```
//PRINT2 EXEC PGM=DSN1LOGP
//STEPLIB  DD DSN=DSN810.SDSNLOAD,DISP=SHR
//SYSABEND DD SYSOUT=*
//SYSSUMRY DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//*BSDS     DD DSN=DSNPB11.PB11.BSDS01,DISP=SHR
//ACTIVE1  DD DISP=SHR,DSN=DSNPB01.PB11.LOGCOPY1.DS02
//ACTIVE2  DD DISP=SHR,DSN=DSNPB01.PB11.LOGCOPY1.DS03
//ACTIVE3  DD DISP=SHR,DSN=DSNPB01.PB11.LOGCOPY1.DS01
//SYSIN    DD *
 DBID(103) OBID(4)
```

列出的 ACTIVE LOG 文件必须按照 RBA 顺序排列，否则会出错。LOG 文件的顺序可以通过 DSNJU004 确定。

3. 打印 ARCHIVE LOG 信息（没有可用的 BSDS 文件）

```
//STEP1 EXEC PGM=DSN1LOGP
//STEPLIB  DD DSN=DSN810.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//ARCHIVE  DD DISP=SHR,
//            DSN=DSNPB01.PB11.ARCLG1.D07360.T1604270.A0006264
//          DD DISP=SHR,
//            DSN=DSNPB01.PB11.ARCLG1.D07358.T2052537.A0006242
```

```
//          DD DISP=SHR,
//             DSN=DSNPB01.PB11.ARCLG1.D07355.T1425469.A0006207
//SYSIN    DD *
  URID(00036B2E7658)
/*
```

4. 在 DATASHARING 环境下对整个 GROUP 使用 DSN1LOGP

```
//STEP1 EXEC PGM=DSN1LOGP
//STEPLIB DD  DSN=DSN710.SDSNLOAD,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSABEND DD  SYSOUT=*
//GROUP DD DSN=DSNPB01.PB11.BSDS01,DISP=SHR
//SYSIN DD *
    DATAONLY (YES)
    LRSNSTART (BB9B81820091) LRSNEND (BB9B818215D6)
/*
```

在 DATASHARING 环境下必须使用 LRSN，不能使用 RBA，而且 LRSN 必须指定 LOG 中存在的值，否则会出错。

5. 在 DATASHARING 环境下对单个的 MEMBER 的 BSDS 做 DSN1LOGP

```
//STEP1 EXEC PGM=DSN1LOGP
//STEPLIB DD  DSN=DSN710.SDSNLOAD,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSABEND DD  SYSOUT=*
//M01BSDS DD DSN=DSNPB01.PB11.BSDS01,DISP=SHR
//SYSIN DD *
  DATAONLY (YES)
  LRSNSTART (BB9B81820091) LRSNEND (BB9B818215D6)
```

4.3.4 DSN1PRNT Utility

4.3.4.1 Utility功能简介

DSN1PRNT 的用途：

- 打印 TABLE，SPACE 和 INDEX SPACE 的 VSAM 文件（包括 DICTIONARY PAGE）；
- 打印 IMAGE COPY 文件；
- 打印包含 TABLESPACE 或 INDEX SPACE 的顺序文件（例如 DSN1COPY 的输出）。

DSN1PRNT 能够打印出 DB2 文件的 16 进制 DUMP。如果指定了 FORMAT 选项，则

DSN1PRNT 对没有错误的 PAGE 进行格式化打印，如果检测到某个页上有错误，则 DSN1PRNT 将错误信息输出，然后继续格式化打印后面的页。

压缩类型的数据需要按照压缩格式打印。

4.3.4.2　控制语句和语法

【语法】请参考《DB2 UDB for z/OS V8 Utility Guide and Reference》。

【主要控制语句描述】

- **32K：**表示输入文件为 32KB 页面，推荐使用 PAGESIZE(32K)。
- **PAGESIZE：**指定 SYSUT1 文件的 PAGE SIZE。可选值为 4KB，8KB，16KB 和 32KB。
- **DSSIZE(integer G)**：指定输入文件的大小，单位是 GB。如果忽略该参数，则 DB2 认为该文件的大小为 2GB；如果是 LOB 的，则 DB2 认为该文件为 4GB。
- **LARGE**：表示输入文件是定义为 LARGE 的 TABLESPACE 或是该 TABLESPACE 上的 INEX。
- **LOB：**指定输入文件是 LOB 类型的 TABLESPACE。
- **NUMPARTS(integer)**：指定输入文件的 PARTITION 的数量。如果输入文件为 PARTITION TABLESPACE，则该参数是必须要写的。有效值为 1～254。该参数帮助 DSN1PRNT 定位要打印的第一个 PAGE。如果忽略该参数或指定为 0，则 DSN1PRNT 认为该 TABLESPACE 是不分 PARTITION 的。如果 NUMPARTS 超过 64，则 DSN1PRNT 认为该 TABLESPACE 是定义为 LARGE 的 PARTITION TABLESPACE。DSN1PRNT 并不能总是验证 NUMPARTS 是否正确，如果指定的参数不正确，则 DSN1PRNT 有可能会出错。
- **PRINT（hexdecimal-contant,hexdecimal-constant）**：指定打印的范围。hexdecimal-contant 指定起始和结束的 PAGE NUMBER。如果没有指定范围，将打印出所有的页。
- **PIECESIZ(integer)：**指定 NON-PARTITIONED INDEX 文件的 PIECE 大小。非 LARGE TABLESPACE 的 INDEX 的默认值为 2GB，LARGE TABLESPACE 的 INDEX 默认值为 4G（4GB）。PIECESIZ 的取值单位有 K（表示 **integer** 的值是乘以 1KB 的结果，**integer** 必须是 256 或 512），M（表示 **integer** 的值是乘以 1MB 的结果，**integer** 必须是 1 到 512 之间的 2 的倍数），G（表示 **integer** 的值是乘以 1GB 的结果，**integer** 必须是 1、2 或 4）。有效的 PIECESIZ 值有：1MB 或 1GB；2MB 或 2GB；4MB 或 4GB；8MB；16MB；32MB；64MB；128MB；256KB 或 256MB；512KB 或 512MB。

- **VALUE：**指定一个字符串值，在 SYSUT1 指定的文件的每个 PAGE 中搜索扫描该值，所有包含该字符串的 PAGE 将被打印出来。该字符串为 1～20 位的数字+字母。

例如：

```
//STEP1 EXEC PGM=DSN1PRNT, PARM='VALUE(''F1F2F3F4F5'')'
```

- **FORMAT：**将输出内容进行格式化。PAGE 中的 CONTROL 信息和 INDIVIDUAL 记录被打印。空的区域则不会显示。
- **FULLCOPY：**指定输入文件是 FULL IMAGE COPY 文件（不是 DFSMS CONCURRENT COPY）。如果被备份的文件是 PARTITION 的，那么必须指定 NUMPARTS 参数。如果省略该参数 FULLCOPY 而输入文件确实是 IMAGE COPY 文件，则 DSN1PRNT 会报错。
- **INCRCOPY：**指定输入文件为 INCREMENTAL COPY 文件。如果被备份的文件是 PARTITION 的，那么必须指定 NUMPARTS 参数。如果省略该参数 INCRCOPY 而输入文件确实是 IMAGE COPY 文件，则 DSN1PRNT 会报错。
- **INLCOPY：**指定输入文件为 INLINE COPY 文件。

4.3.4.3 如何运行DSN1PRNT

1. 作业中需要的文件

- SYSPRINT：存放输出信息和 16 进制 DUMP 输出的文件。
- SYSUT1：输入文件，可以是 VSAM 文件和 PS 文件。DSN1PRNT 认为该文件的块大小为 4096 字节（DB2 文件标准）。DISPOSITION 要指定为 OLD 以确定该文件不被 DB2 使用；当使用了 STOP DATABASE 命令停掉相应的 DATABASE 后可以使用 DISP=SHR。

2. 使用 DSN1PRNT 取代 DSN1COPY

如果要打印文件的信息，最好用 DSN1PRNT 代替 DSN1COPY。因为 DSN1COPY 会扫描整个文件，而 DSN1PRNT 只会扫描指定的范围，并且 DSN1PRNT 也能打印出格式化的 DUMP 信息。

3. 查看 PAGE SIZE 和 DSSIZE

用下面的 SQL 语句查看文件的 PAGE SIZE 和 DSSIZE：

```
SELECT I.CREATOR,
    I.NAME,
    S.PGSIZE,
```

```
    CASE S.DSSIZE
      WHEN 0 THEN CASE S.TYPE
                  WHEN ' ' THEN 2097152
                  WHEN 'I' THEN 2097152
                  WHEN 'L' THEN 4194304
                  WHEN 'K' THEN 4194304
                  ELSE NULL
                END
      ELSE S.DSSIZE
    END
FROM SYSIBM.SYSINDEXES I,
    SYSIBM.SYSTABLES  T,
    SYSIBM.SYSTABLESPACE S
WHERE I.CREATOR='DSN8610' AND
     I.NAME='XEMP1' AND
     I.TBCREATOR=T.CREATOR AND
     I.TBNAME=T.NAME AND
      T.DBNAME=S.DBNAME AND
     T.TSNAME=S.NAME;
```

4.3.4.4　参考作业流

1. 打印一个非 PARTITION TABLESPACE 的信息

```
//RUNPRNT EXEC PGM=DSN1PRNT,PARM='PRINT(0,1000),FORMAT'
//STEPLIB DD DSN=DSN710.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD  DISP=SHR,
//          DSN=DSNZB0B.DSNDBC.TEMPDB.TEMPTS.I0001.A001
```

2. 打印 PARTITION TABLESPACE 的信息

```
//RUNPRNT EXEC PGM=DSN1PRNT,PARM='PRINT(0,1000),FORMAT,
//             NUMPARTS(4)'
//STEPLIB DD DSN=DSN710.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DISP=SHR,
//         DSN=DSNZB0B.DSNDBC.BCASDB.BSARRTD.I0001.A001
```

由于篇幅有限，本章只列出主要的 DB2 常用 Utility 的使用方法举例，其他 Utility 的详细语法和使用限制可以参考《DB2 UDB for z/OS V8 Utility Guide and Reference》。



4.4 课后习题

1．对于数据库表的备份通常使用什么 Utility？与 UNLOAD Utility 相比哪个速度更快一些？

2．如果表空间效率低下，使用什么 Utility 来帮助提高访问性能？

3．RUNSTATS Utility 主要作用是什么？什么情况下使用？

4．有哪几种情况出现必须使用 REBULID INDEX Utility？

5．如何控制正在运行的 ONLINE Utility？

6．表空间 SYSUTILX 在 ONLINE Utility 运行过程中起到了什么作用？

第 5 章　DB2 常用命令

本章主要介绍 DB2 的基本命令，涉及启下系统、检查 DB2 运行状态、检查数据库对象状态、解决异常状态等操作。



5.1　DB2 命令介绍

5.1.1　DB2 命令的作用范围

一些命令在 Data Sharing 环境中发出后只在当前 MEMBER 中有作用，例如，CANCEL THREAD 只对当前 MEMBER 中的 THREADS 作用。另外一些命令发出后影响 Data Sharing 中的所有 MEMBER。

作用范围是 GROUP 的命令包括：

```
ALTER GROUPBUFFERPOOL          DISPLAY DATABASE
DISPLAY GROUP                  DISPLAY GROUPBUFFERPOOL
MODIFY irlmproc,DIAG           START DATABASE
STOP DATABASE
```

下面的命令根据命令参数决定作用范围是 GROUP 还是 MEMBER：

```
ARCHIVE LOG                    DISPLAY FUNCTION SPECIFIC
DISPLAY PROCEDURE              DISPLAY THREAD
```

```
DISPLAY TRACE                  DISPLAY Utility
MODIFY irlmproc,SET            MODIFY irlmproc,STATUS
START FUNCTION SPECIFIC        START PROCEDURE
START TRACE                    STOP FUNCTION SPECIFIC
STOP PROCEDURE                 STOP TRACE
TERM Utility
```

5.1.2 DB2 命令的提交方式

DB2 命令可以在以下几种环境中提交：

- z/OS console 控制台
- 应用程序
- DSN 的 Session
- DB2I 交互界面
- IMS 的终端
- CICS 的终端
- 使用 DB2 instrumentation facility interface（IFI）的应用程序
- TSO SDSF

通过登录到 z/OS console 或 TSO SDSF 提交 DB2 命令的用户通过 DB2 权限控制检查确认该 ID 是否有权限提交此命令。

最常用的方法是在 DB2I 或 TSO SDSF 中提交，这样可以直接看到命令的返回信息，方便用户检查确认命令执行情况。

5.1.3 DB2 命令的使用方法

DB2 命令的一般格式如下：

```
命令前缀（Command Prefix） + 具体命令（command）
```

其中命令前缀是在操作系统参数 IEFSSNxx 中定义的，每个 DB2 MEMBER 都有一个对应的命令前缀。

举例说明：

```
-DPR1 DISPLAY THREAD
```

此命令的效果是显示 DB2 MEMBER DPR1 上当前的线程信息，其中“-DPR1”是命令前缀，“DISPLAY THREAD”则是具体的 DB2 命令。

5.2　DB2 系统相关命令

5.2.1　–START DB2

–START DB2 启动一个 DB2 子系统，对于 Data Sharing 环境必须先启动第一个 MEMBER。这个命令只可以在 z/OS console 发出，命令缩写为：-STA DB2，常用参数包括：PARM，ACCESS。

语法图如图 5.1 所示。

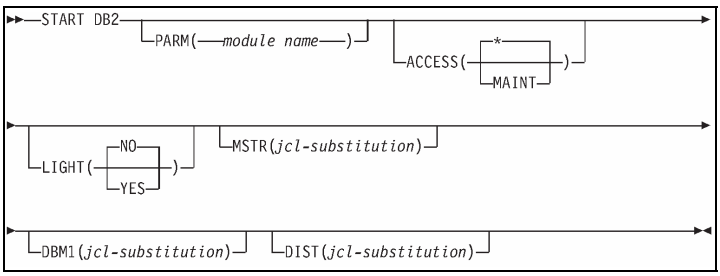


图 5.1　–START DB2 语法图

参数说明如下。

ACCESS

- *：DB2 启动后允许所有的授权用户连接
- MAINT：DB2 系统启动后只有安装时指定的 SYSADM 和 SYSOPR 用户才可以连接 DB2（也就是在 DSNZPARM 参数 SYSADMS 和 SYSOPRS 指定的两个用户），在 Data Sharing 中只有当前启动的 MEMBER 才受到这个参数的影响。
- 使用命令：/PT11 –PB11 START DB2，可在 PT11 上启动 PB11 这个子系统。
- 使用命令：/PT11 –PB11 START DB2 ACCESS（MAINT），可在 PT11 上启动 PB11 这个子系统，系统启动后只有安装时指定的 SYSADM 和 SYSOPR 用户才可以连接 DB2（也就是在 DSNZPARM 参数中 SYSADMS 和 SYSOPRS 指定的两个用户），在 Data Sharing 中只有当前启动的 MEMBER 才受到这个参数的影响。

5.2.2 –STOP DB2

–STOP DB2 停止一个 DB2 子系统。

命令缩写为：-STO DB2，常用参数包括：MODE,CASTOUT。

语法图如图 5.2 所示。

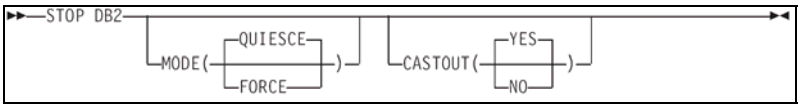


图 5.2 –STOP DB2 语法图

- 使用命令：-PB11 STOP DB2 可以将 PB11 停止。执行停止 DB2 的命令需要下面的权限之一：STOPALL、SYSOPR、SYSCTRL、SYSADM。
- 使用命令：-PB11 STOP DB2 MODE (QUIESCE) CASTOUT(YES)等当前运行在 PB11 上的程序和事务结束后，允许 GBP 中的数据写回到磁盘中（如果不需要 GBP 的数据写回磁盘，可以用参数 CASTOUT（NO)），再停止 PB11 子系统，但是不允许新的程序和新的数据库连接。
- 使用命令：-PB11 STOP DB2 MODE (FORCE)立即终止所有的程序和 Utility，停止 DB2 子系统。但是这种方式会使 DB2 重启后有些问题，一些线程将处于 indoubt 状态，一些表和 Utility 的状态将不正常。所以慎用这种方式停止 DB2。
- 使用命令：/CANCEL PB11，也可以强制停掉 DB2。

5.2.3 –SET SYSPARM

–SET SYSPARM 修改 DB2 启动参数，语法图如图 5.3 所示。

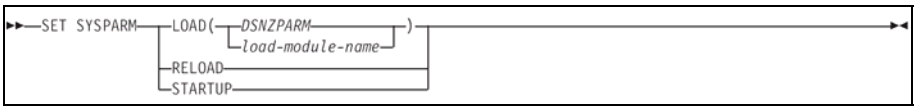


图 5.3 –SET SYSPARM 语法图

参数说明如下。

1．LOAD(*load-module-name*)

指定 DB2 启动时默认的 LOAD MODULE，默认的 LOAD MODULE 是 DSNZPARM。

2．RELOAD

重新 LOAD 上次设定的 LOAD MODULE。

3．STARTUP

当 DB2 重启的时候重新 LOAD 上次的 LOAD MODULE。

- 使用命令：-SET SYSPARM LOAD(ADMPARM1) 改变 LOAD MODULE 为 ADMPARM1，如果以前的 LOAD MODULE 是 DSNZPARM，这个命令将 LOAD MODULE 改为 ADMPARM1。
- 使用命令：-SET SYSPARM RELOAD 重新载入当前用的 LOAD MODULE ADMPARM1。
- 使用命令：-SET SYSPARM STARTUP 当 DB2 重启的时候重新 LOAD ADMPARM1。

5.2.4　–DISPLAY GROUP

–DISPLAY GROUP 显示 DB2 MEMBER 所在的 group 的信息，包括 SCA 的大小及使用情况、每个 MEMBER 连接的 IRLM 子系统的名字和 IRLMPROC 的名字。语法图如图 5.4 所示。

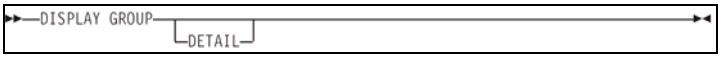


图 5.4　–DISPLAY GROUP 语法图

其中每个 MEMBER 的状态可以有三种情况：ACTIVE、QUIESCED 和 FAILED。ACTIVE 指正常可用的；QUIESCED 表示这个 MEMBER 处于正常的停止状态，是执行 STOP DB2 命令后的正常状态；FAILED 表示这个 MEMBER 处于异常下宕状态，需要用户及时处理。

- 使用命令-PB11 DIS GROUP 后信息显示如下：

```
--PB11 DIS GROUP
DSN7100I  -PB11 DSN7GCMD
*** BEGIN DISPLAY OF GROUP(DSNPB01 ) GROUP LEVEL(810) MODE(N)
                  PROTOCOL LEVEL(2)  GROUP ATTACH NAME(PB01)
------------------------------------------------------------------------
DB2                                        DB2 SYSTEM     IRLM
MEMBER    ID  SUBSYS CMDPREF    STATUS    LVL NAME       SUBSYS IRLMPROC
--------  --- ----- --------    --------  --- --------   ----   --------
PB11        1 PB11  -PB11       ACTIVE    810 PT11       PJ11   PB11IRLM
PB21        2 PB21  -PB21       ACTIVE    810 PT12       PJ21   PB21IRLM
PB31        3 PB31  -PB31       ACTIVE    810 PT13       PJ31   PB31IRLM
PB41        4 PB41  -PB41       ACTIVE    810 PT14       PJ41   PB41IRLM
PB51        5 PB51  -PB51       QUIESCED  810 PT11       PJ51   PB51IRLM
PB61        6 PB61  -PB61       QUIESCED  810 PT12       PJ61   PB61IRLM
PB71        7 PB71  -PB71       QUIESCED  810 PT13       PJ71   PB71IRLM
PB81        8 PB81  -PB81       QUIESCED  810 PT14       PJ81   PB81IRLM
------------------------------------------------------------------------
SCA   STRUCTURE SIZE:    24576 KB, STATUS= AC,   SCA IN USE:      1 %
LOCK1 STRUCTURE SIZE:  1258496 KB
NUMBER  LOCK ENTRIES:   134217728
NUMBER  LIST ENTRIES:     2843791, LIST ENTRIES  IN USE:        5198
*** END DISPLAY OF GROUP(DSNPB01 )
DSN9022I  -PB11 DSN7GCMD 'DISPLAY GROUP ' NORMAL COMPLETION
```

5.2.5 –RECOVER BSDS

语法图如图 5.5 所示。

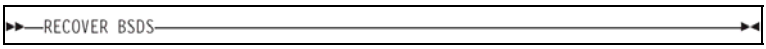


图 5.5 –RECOVER BSDS 语法图

当 BSDS 中的一个文件发生错误时，重新定义一个 BSDS 文件后，用这个命令来恢复双工的 BSDS 数据集。必须用下面的步骤来恢复双工 BSDS：

- 删除或者重命名发生错误的 BSDS 数据集，用相同的名字重新定义一个新的 BSDS 数据集，可以参照安装库中的 DSNTIJIN 作业来定义。
- 用这个命令恢复双工的 BSDS，同时没有发生错误的那一个 BSDS 会将它的内容复制到新定义的 BSDS 文件中。

5.2.6 –RECOVER INDOUBT

–RECOVER INDOUBT 恢复 INDOUBT THREAD。缩写为-REC IND。

语法图如图 5.6 所示。

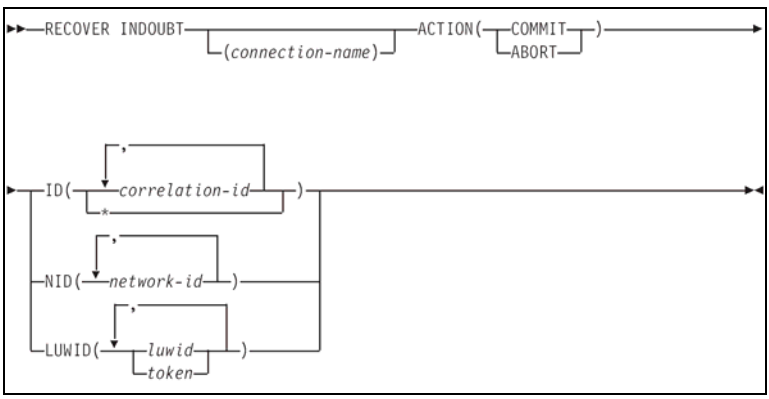


图 5.6 –RECOVER INDOUBT 语法图

- 使用命令：

```
-PB11 RECOVER INDOUBT(CONNID) ID(CORRELATIONID) ACTION(XX)
```

来恢复 INDOUBT THREAD，其中 ACTION 的选项有 COMMIT 和 ABORT，COMMIT

表示提交这个 THREAD，ABORT 表示 CANCEL 这个 THREAD，一般选择 ABORT。

5.3　DATABASE 相关命令

5.3.1　–ACCESS DATABASE

–ACCESS DATABASE 打开指定的表空间或者索引空间，也可以通过用这个命令来移除表空间的 GBP-dependent 状态。

语法图如图 5.7 所示。

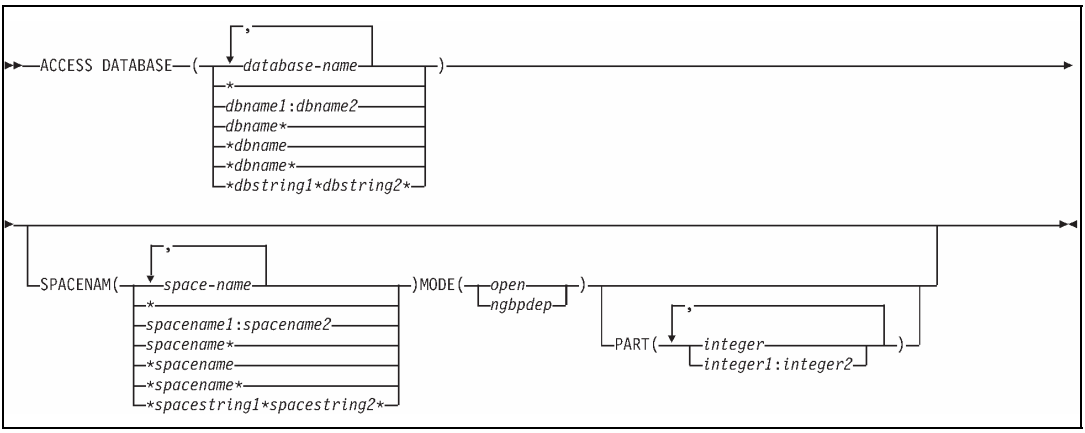


图 5.7　–ACCESS DATABASE 语法图

参数说明如下。

- **Mode (OPEN)**：物理地打开表空间。
- **Mode (NGBPDEP)**：移除表空间的 GBP-dependent 状态。

 使用命令：

```
- ACCESS DATABASE(DSN9001) SPACENAM(DSN9003) MODE(NGBPDEP)
```

关闭表空间 DSN9003，并且使它不处于 GBP-dependent 的状态。

5.3.2　–START DATABASE

–START DATABASE 命令可以用来启动 DB、表空间、索引空间等，缩写为-STA DB，

常用参数包括：SPACENAM（缩写为 SP），PART，ACCESS（缩写为 ACC）等。

语法图如图 5.8 所示。

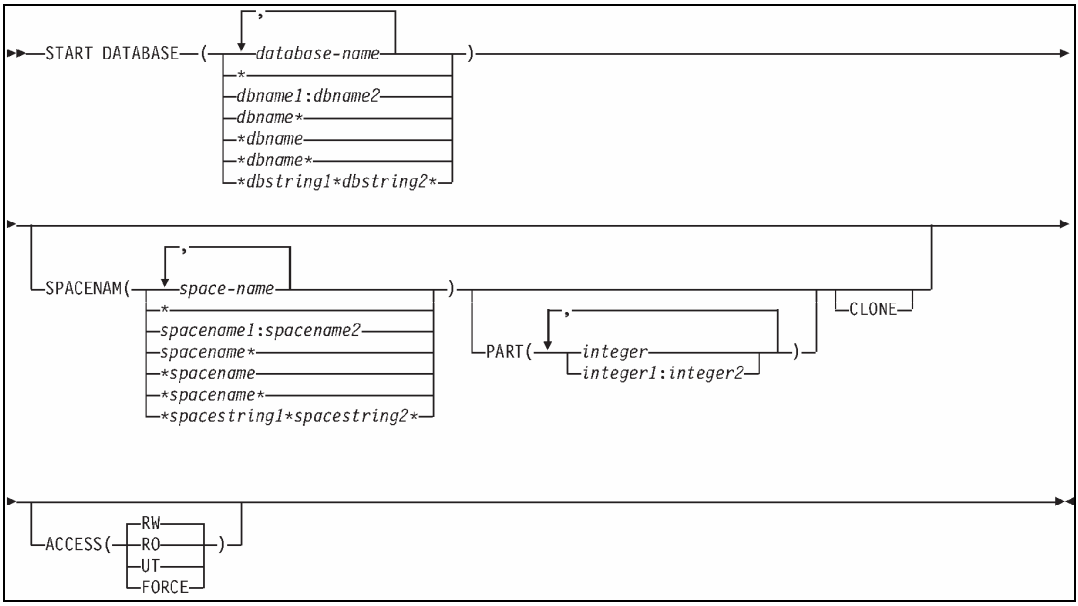


图 5.8 –START DATABASE 语法图

参数说明如下。

ACCESS：指定数据库或表空间被启动后的访问状态，可读可写，只读等。

RW：可读可写，这是最正常的访问状态。

RO：只读状态。

UT：只允许 Utility 和 DROP 语句访问。

FORCE：可以重置 LPL，read-only accesses，Utility accesses，CHECK-pending, COPY-pending 和 RECOVER-pending 的状态，但是 FORCE 不能用来重置 restart-pending (RESTP)状态。

- 使用命令：-PB11 STA DB(*)启动 PB11 的所有 DATABASE。
- 使用命令：-PB11 STA DB(NASEDB)只启动 PB11 的 NASEDB。
- 使用命令：-PB11 STA DB(NAS*)启动 PB11 中的以 NAS 开头的所有 DB。
- 使用命令：-PB11 STA DB(NASEDB) SP（SPANME）RO 以只读模式启动 NASEDB 中的 SPNAME，启动模式还包括 UT、FORCE、RW。

5.3.3　–STOP DATABASE

–STOP DATABASE 命令可以用来停止 DB、表空间、索引空间等，缩写为-STO DB，语法图如图 5.9 所示。

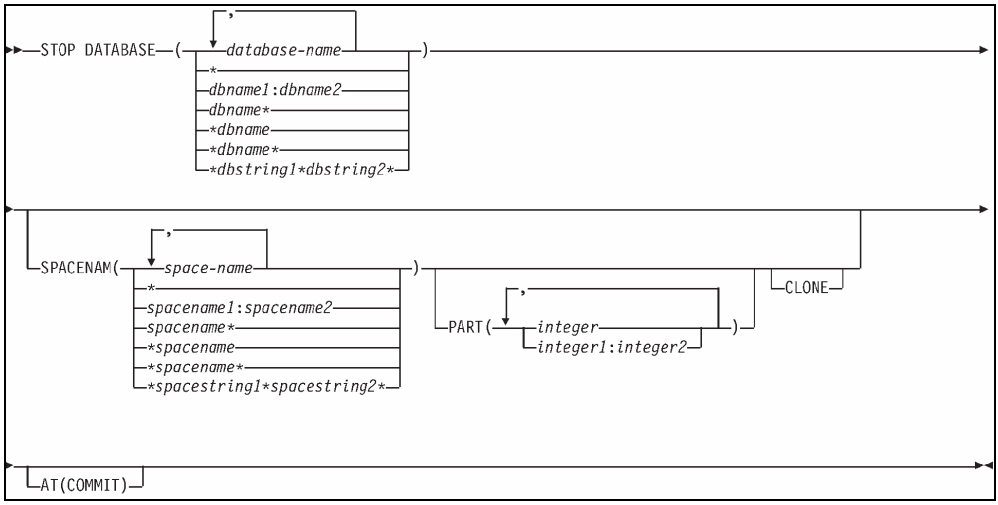


图 5.9　–STOP DATABASE 语法图

参数说明如下。

AT(COMMIT)将指定对象 STOP，并且阻止新的访问请求，对于正在访问运行的程序，其当前的下一个 commit 之后的访问请求也会被阻止。

- 使用命令：-PB11 STO DB(*)停止 PB11 的所有 DATABASE。
- 使用命令：-PB11 STO DB(NASEDB)只停止 PB11 的 NASEDB。
- 使用命令：-PB11 STO DB(NAS*)停止 PB11 中的以 NAS 开头的所有 DB。
- 使用命令：-PB11 STO　DB(NASEDB) SP（SPNAME）AT(COMMIT)停止 NASEDB 中的 SPNAME，AT(COMMIT) 指等正在运行的程序和 SQL 到了下一个 commit 点的时候再停止表空间，但是不允许新的连接请求。

5.3.4　–DISPLAY DATABASE

–DISPLAY DATABASE 查看数据库和表空间的状态信息，缩写为-DIS DB，语法图如图 5.10 所示。

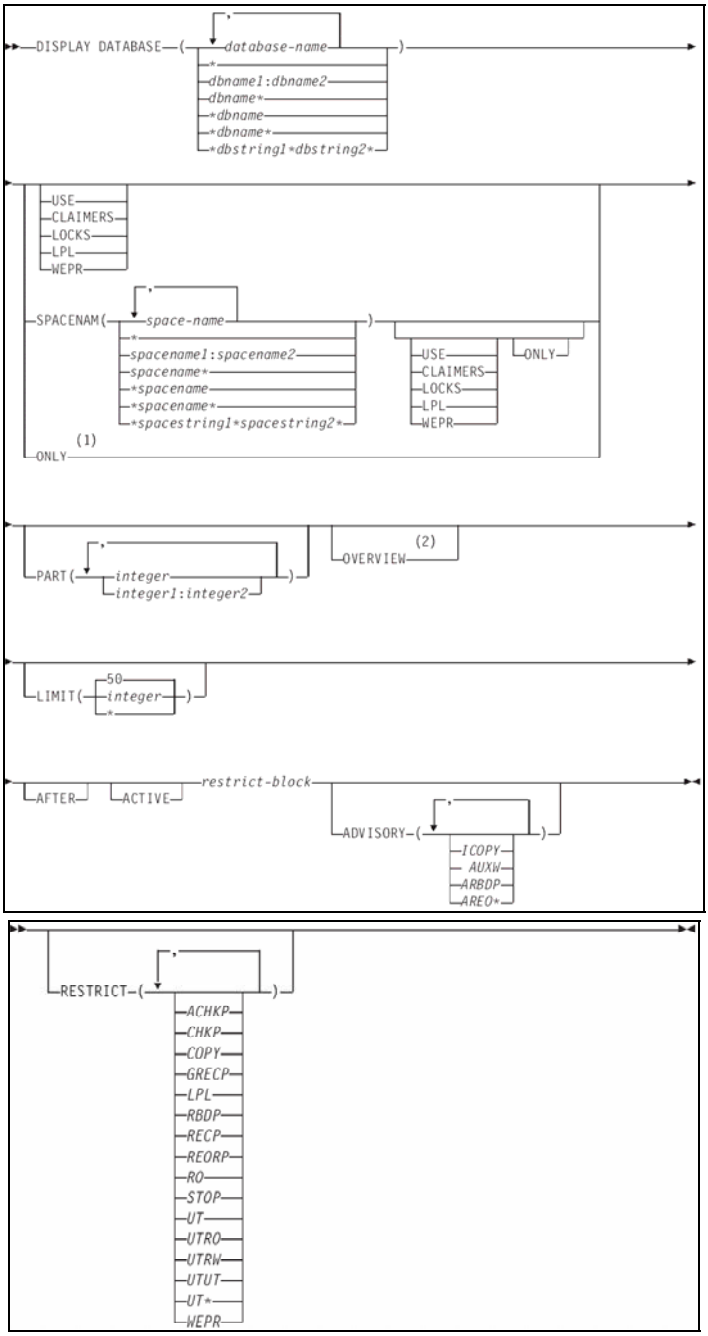


图 5.10 – DISPLAY DATABASE 语法图

- 使用命令：

```
-DISPLAY DATABASE(CB3) SPACENAM(TBS33) USE
```

显示数据库 CB3 中的 TBS33 的状态。USE 参数可以显示 CONNID、CORRID 和 USERID。

CONNID 表示连接类型，显示线程是从哪种方式连接上来的，可以是如下几种：BATCH、TSO、Utility、IMS identifier、CICS identifier、CONSOLE。

CORRID 表示这个连接的所属 ID，如果 CONNID 是 BATCH 那么 CORRID 就是作业名；如果 CONNID 是 TSO，那么 CORRID 就是登录 TSO 的用户 ID；如果 CONNID 是 CICS identifier，那么 CORRID 就是接入 ID 线程号。

- 使用命令：

```
-DB1G DISPLAY DATABASE(DSN8D81A) SPACE(TSPART) LOCKS
```

显示指定表空间的锁信息。

主要显示信息是：

```
NAME,TYPE,PART,STATUS,CONNID,CORRID,LOCKINFO
```

TYPE——TS 表示表空间，IX 表示索引空间，TB 表示表。

LOCKINFO——表示锁的类型和锁持续时间。格式是：LOCK STATUS, LOCK STATE,LOCK TYPE,LOCK DURATION。

- 使用命令：

```
-DISPLAY DATABASE(CB3) SPACENAM(TBS33) CLAIMERS
```

显示表空间上的 CLAIM 类型和持续时间。

- 使用命令：

```
- DISPLAY DATABASE(DSNDB01) SPACENAM(*) LIMIT(*) LPL
```

显示数据库 DSNDB01 中在 LPL 的页，LIMIT 是限制命令的输出行数，用*指不限制输出行数。

- 使用命令：

```
-DISPLAY DATABASE(DBKD0101,DBKD0103) SPACENAM(*) RESTRICT LIMIT(*)
```

显示从数据库 DBKD0101 到 DBKD0103 的处于 RESTRICTIVE 状态的所有表空间和索

引空间。

- 使用命令：

```
--PB11 DIS DB(*) SP(*) USE
```

显示当前哪些应用在使用哪些表空间资源。信息显示如下：

```
DSNT360I  -PB11 ************************************
DSNT361I  -PB11 *  DISPLAY DATABASE SUMMARY
                *     GLOBAL USE
DSNT360I  -PB11 ************************************
DSNT362I  -PB11      DATABASE = DSNDB01  STATUS = RW
                   DBD LENGTH = 14200
DSNT397I  -PB11
NAME     TYPE PART  STATUS              CONNID   CORRID       USERID
-------- ---- ----- ------------------ -------- ------------ --------
DBD01    TS         RW
SPT01    TS         RW                  BATCH    OFBJ1807     NGTNS
   -                   MEMBER NAME PB31
SPT01    TS         RW                  BATCH    OFBJBTH5     NGTNS
   -                   MEMBER NAME PB41
SPT01    TS         RW                  BATCH    OFBJBTH4     NGTNS
   -                   MEMBER NAME PB41
SPT01    TS         RW                  BATCH    CBP0807      NGTNS
   -                   MEMBER NAME PB11
SCT02    TS         RW
```

- 使用命令：

```
-DIS DB(MVIMSDB) SP(MSBDREAL) LOCKS
```

显示表空间 MSBDREAL 上的 LOCK 信息显示如下：

```
DSNT360I  -PB11 ************************************
DSNT361I  -PB11 *  DISPLAY DATABASE SUMMARY
                *     GLOBAL LOCKS
DSNT360I  -PB11 ************************************
DSNT362I  -PB11      DATABASE = MVIMSDB  STATUS = RW
                   DBD LENGTH = 395714
DSNT397I  -PB11
NAME     TYPE PART  STATUS              CONNID   CORRID       LOCKINFO
-------- ---- ----- ------------------ -------- ------------ ---------
MSBDREAL TS         RW                                       H-S,PP,I
   -                   MEMBER NAME PB31
******** DISPLAY OF DATABASE MVIMSDB  ENDED      **********************
DSN9022I  -PB11 DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION
```

5.4 BP 和 GBP 相关命令

5.4.1 –ALTER BUFFERPOOL

–ALTER BUFFERPOOL 调整活动的和不活动的 BUFFERPOOL 的属性。BUFFERPOOL 的信息都写在 BSDS 中。

语法图如图 5.11 所示。

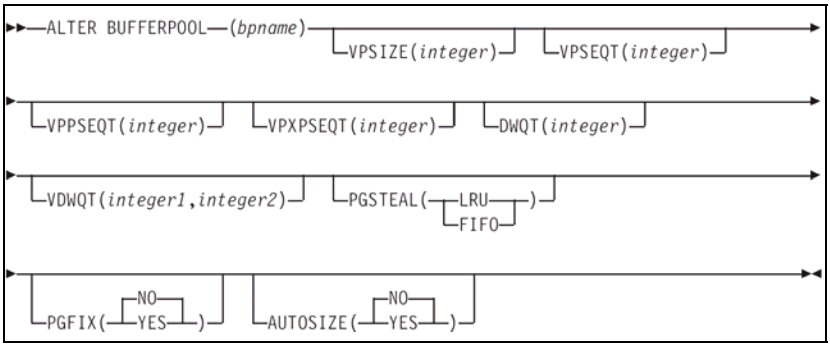


图 5.11　–ALTER BUFFERPOOL 语法图

参数说明如下。

1．BPNAME。

4-kb 页面的 bufferpool 命名为：BP0，BP1，…，BP49。

8-kb 页面的 bufferpool 命名为：BP8K0，BP8K1，…，BP8K9。

16-kb 页面的 bufferpool 命名为：BP16K0，BP16K1，…，BP16K9。

32-kb 页面的 bufferpool 命名为：BP832K0，BP32K1，…，BP32K9。

2．VPSIZE (integer)

是设定的 bufferpool 的大小。

除去 bp0 以外的 4-kb 页面的 bufferpool 的大小范围是 0～250 000 000。

BP0 bufferpool 的大小范围是 2000～250 000 000。

8-kb 页面的 bufferpool 的大小范围是 1000～125 000 000。

16-kb 页面的 bufferpool 的大小范围是 0～62 500 000。

32-kb 页面的 bufferpool 的大小范围是 0～31 250 000。

3．VPSEQT (integer)

设置 steal 的置换阈值。

4．PGSTEAL

设定 bufferpool 的置换算法。

LRU：最近最久未使用算法，是默认值。

FIFO：先进先出算法。

5．AUTOSIZE

设定是否允许 WLM 自动调整 BUFFERPOOL 的大小。

- 使用命令：

```
-ALTER BUFFERPOOL(bpname) VPSIZE(0)
```

Deallocate 这个 bufferpool，也就是删除这个 bufferpool。

5.4.2 –ALTER GROUPBUFFERPOOL

–ALTER GROUPBUFFERPOOL 调整 GROUPBUFFERPOOL 的属性。

语法图如图 5.12 所示。

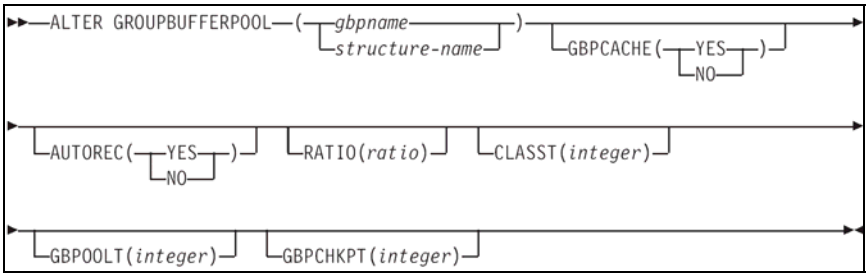


图 5.12 –ALTER GROUPBUFFERPOOL 语法图

参数说明如下。

1．GBPNAME

4-kb 页面的 group bufferpool 命名为：GBP0，GBP1，…，GBP49。

8-kb 页面的 group bufferpool 命名为：GBP8K0，GBP8K1，…，GBP8K9。

16-kb 页面的 group bufferpool 命名为：GBP16K0，GBP16K1，…，GBP16K9。

32-kb 页面的 group bufferpool 命名为：GBP832K0，GBP32K1，…，GBP32K9。

GBPCACHE 指 GBP 是否用来做数据 CACHE。

YES 指 GBP 用来作为数据 CACHE，也用来做 cross-invalidation。

NO 指 GBP 只用来做 cross-invalidation。

2．GBPOOLT（integer）

改变 GBP 中数据写回到硬盘上的阈值，范围是 0～90%，默认值是 30%。例如 GBPOOLT

（55），即当 GBP 中的数据 page 数超过 GBP 全部 page 数的 55%时就开始把这些数据写回到磁盘上去。

3．GBPCHKPT(integer)

以分钟计，设置 GBP 作为 checkpoint 的时间间隔。默认值是 4 分钟。Checkpoint 做得越频繁，当 CF 出现问题的时候 GBP 恢复所需要的时间也就越短。

5.4.3　–DISPLAY BUFFERPOOL

–DISPLAY BUFFERPOOL 查看某个 BUFFERPOOL 的状态和设置，也可指定具体的 BUFFERPOOL 名字。

语法图如图 5.13 所示。

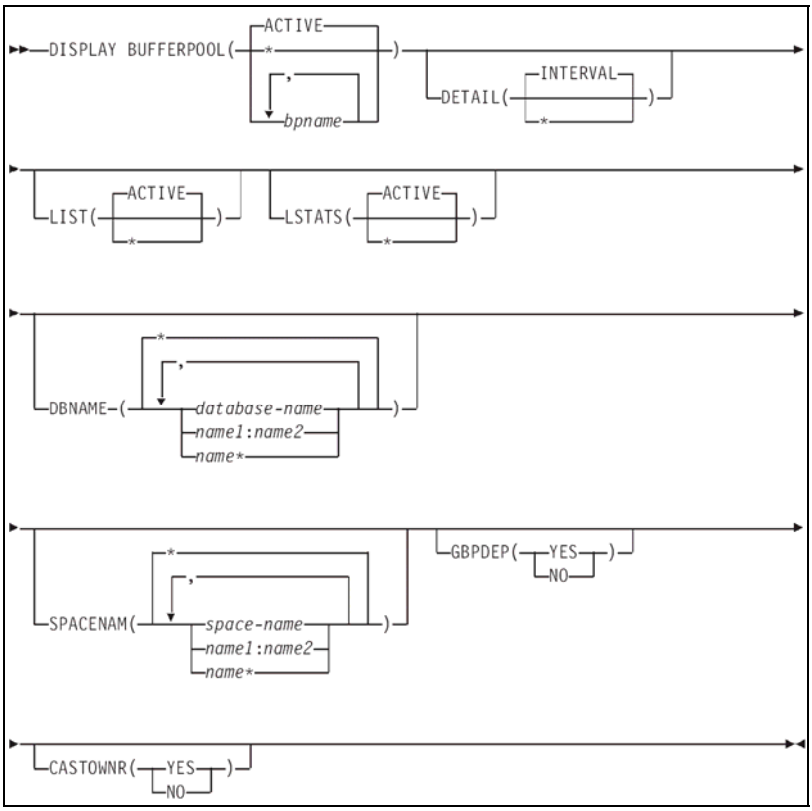


图 5.13　–DISPLAY BUFFERPOOL 语法图

参数说明如下。

1．BUFFERPOOL

ACTIVE：只显示 ACTIVE 状态的 BUFFERPOOL 的状态。

*：显示所有 BUFFERPOOL（包括 ACTIVE 和 INACTIVE）的状态。

bpname：需要查看的 BUFFERPOOL 的名字。

2．DETAIL

对指定的 BUFFERPOOL 产生一个详细信息的报告，如果不指定这个参数，就会只产生一个简单的 SUMMARY REPORT。

3．LIST

列出在这个 BUFFERPOOL 中打开的索引空间和表空间的信息。ACTIVE 只显示在这个 BP 中打开的并且正在被使用的表空间和索引空间的信息。

*：显示在这个 BP 中打开的所有表空间和索引空间，不管是不是正被使用。

4．LSTATS

列出与这些表空间和索引空间有关的 DATA SET 的统计信息。

ACTIVE：限制只显示那些正在被使用的 DATA SET 的统计信息。

*：显示所有的和这些表空间和索引空间有关的 DATA SET 的统计信息。

5．GBPDEP

是否限制 GBP DEPENDENT 的 DATA SET 信息。

YES：限制只显示那些 GBP-dependent 的 DATA SET。

NO：限制只显示那些 NON-GBP-dependent 的 DATA SET。

6．CASTOWNR

限制这个 DB2 MEMBER 是否是这个 DATA SET 的 CASTOUT OWNER。

YES：只显示当前 DB2 MEMBER 是其 CASTOUT OWNER 的 DATA SET 信息。

NO：只显示当前 DB2 MEMBER 不是其 CASTOUT OWNER 的 DATA SET 信息。

- 使用命令：

```
-DISPLAY BUFFERPOOL(BP0) DETAIL
```

显示 BP0 的详细信息，包括 summary report 和相关的统计信息。

- 使用命令：

```
-DIS BUFFERPOOL(BP0) LIST(ACTIVE)
```

显示 BP0 的相关信息及用到 BP0 打开的表空间和索引空间。

- 使用命令：

```
-PB11 DIS BUFFERPOOL(BP0)
```

信息显示如下：

```
--PB11 DIS BUFFERPOOL(BP0)
 DSNB401I  -PB11 BUFFERPOOL NAME BP0, BUFFERPOOL ID 0, USE COUNT 127
 DSNB402I  -PB11 BUFFER POOL SIZE = 5000 BUFFERS
              ALLOCATED          =     5000    TO BE DELETED    =        0
              IN-USE/UPDATED     =        0    BUFFERS ACTIVE   =     5000
 DSNB406I  -PB11 PGFIX ATTRIBUTE -
              CURRENT = NO
              PENDING = NO
            PAGE STEALING METHOD = LRU
 DSNB404I  -PB11 THRESHOLDS -
             VP SEQUENTIAL     = 80
             DEFERRED WRITE    = 50    VERTICAL DEFERRED WRT  = 10,  0
             PARALLEL SEQUENTIAL =50   ASSISTING PARALLEL SEQT=  0
 DSN9022I  -PB11 DSNB1CMD '-DIS BUFFERPOOL' NORMAL COMPLETION
```

5.4.4　–DISPLAY GROUPBUFFERPOOL

–DISPLAY GROUPBUFFERPOOL 查看 GBP 的信息。

语法图如图 5.14 所示。

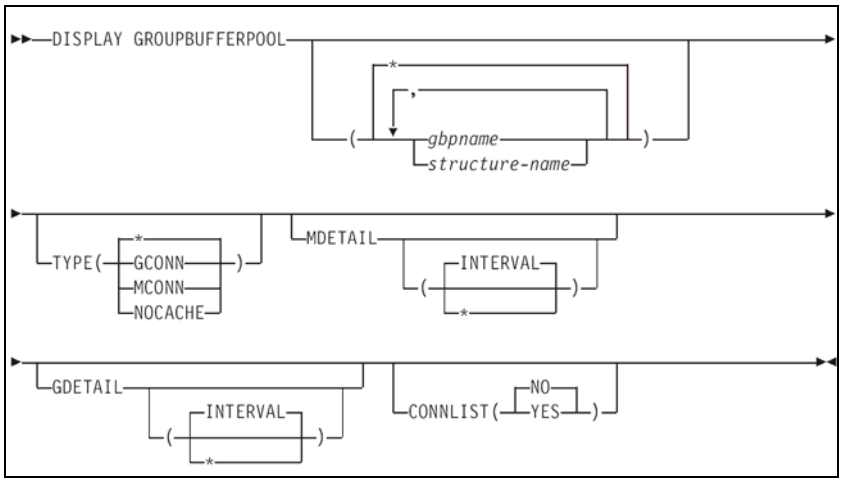


图 5.14　–DISPLAY GROUPBUFFERPOOL 语法图

- 使用命令：

```
-DISPLAY GROUPBUFFERPOOL(*) GDETAIL(INTERVAL)
```

按照一定的 INTERVAL 收集 GBP 统计信息。



5.5 Utility 相关命令

5.5.1 –ALTER Utility

–ALTER Utility 修改正在运行的 rebuild（必须用 shrlevel change 参数）和 reorg（必须用 shrlevel reference 或者 shrlevel change）的 DEADLINE, MAXRO, LONGLOG 和 DELAY 参数。这个命令必须在 rebuild 和 reorg 工具运行的子系统上发出。命令缩写为：

```
- ALT UTIL
```

语法图如图 5.15 所示。

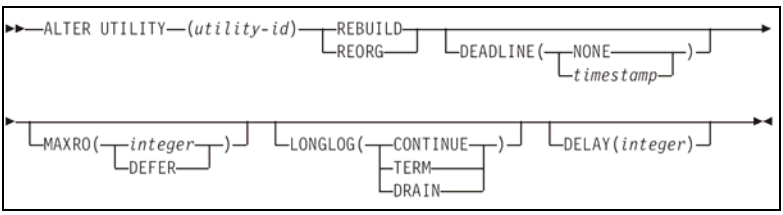


图 5.15 –ALTER Utility 语法图

参数说明如下。

1．utility-id

指定要修改属性的 REORG 或者 REBUILD Utility 的 id 号。

2．DEADLINE

指定 reorg 的时候，其 switch 阶段的 deadline，如果这个时间点前没有开始进行 reorg 的 switch 阶段，那么这个 reorg 就会自己结束，相当于不做 reorg。

NONE：指定 reorg 工具没有 deadline 参数的限制。

timestamp：为 SWITCH 阶段指定 deadline，当发 ALT UTIL 命令的时候这个 deadline 必须还没有到。

3．MAXRO

设置 REORG 的 log 阶段允许的最长时间，在 log 阶段对应的表只能读不能写。

INTEGER：设置 MAXRO 的值，以秒为单位。

DEFER：LOG 阶段被延后。

- 使用命令：

```
-ALTER Utility (REORGEMP) REORG MAXRO(240) LONGLOG(DRAIN)
```

修改正在执行的 REORG Utility(REORGEMP)的 MAXRO 参数为 240 秒，LONGLOG 参数为 DRAIN。

5.5.2　–DISPLAY Utility

–DISPLAY Utility 查看 Utility 状态。

语法图如图 5.16 所示。

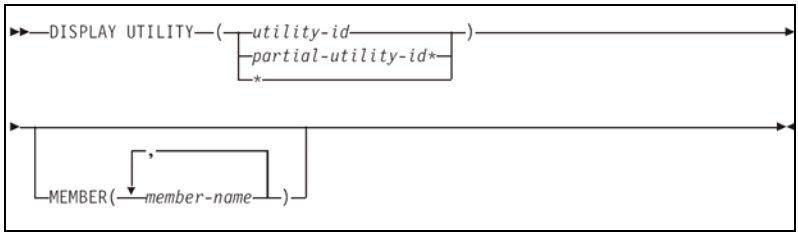


图 5.16　–DISPLAY Utility 语法图

- 使用命令：

```
-PB11 DIS Utility(*)
```

信息显示如下：

```
--PB11 DIS UTILITY(*)
 DSNU112I  -PB11 DSNUGDIS - NO AUTHORIZED UTILITY FOUND FOR UTILID = *
 DSN9022I  -PB11 DSNUGCCC '-DIS UTILITY' NORMAL COMPLETION
```

5.5.3　–TERM Utility

–TERM Utility 终止正在运行的 DB2 Utility 并释放该 Utility 所占用的相关资源，ACTIVE Utility 只可以在它运行的 MEMBER 上 TERM 掉，STOPPED Utility 可以从 GROUP

中的任何一个 MEMBER 上 TERM 掉。缩写为：

```
-TER UTIL
```

语法图如图 5.17 所示。

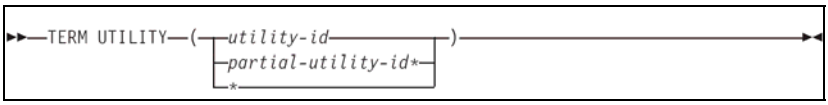


图 5.17　–TERM Utility 语法图

- 使用命令：

```
-TERM Utility(Utility-id)
```

终止正在运行的 DB2 Utility 并释放该 Utility 所占用的相关资源，或是终止已经异常中断的 DB2 Utility。

5.6　TRACE 相关命令

5.6.1　–START TRACE

–START TRACE 启动 DB2 TRACE。缩写为：

```
-STA TRA
```

语法图如图 5.18～5.20 所示。

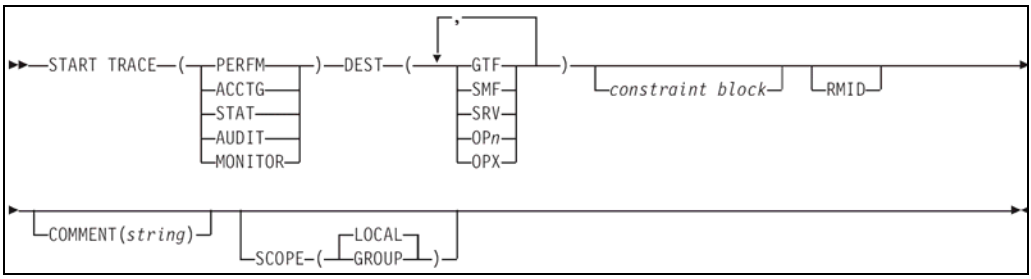


图 5.18　–START TRACE 语法图

- 使用命令：

```
-START  TRACE(XXXX)  CLASS(X,X,X,。。。。)  SCOPE(GROUP)
```

指定 TRACE 的类型包括：PERFM，ACCTG，STAT，AUDIT，MONITOR；CLASS 根据不同的 TRACE 类型和不同的数字来表示 TRACE 跟踪的不同指标。

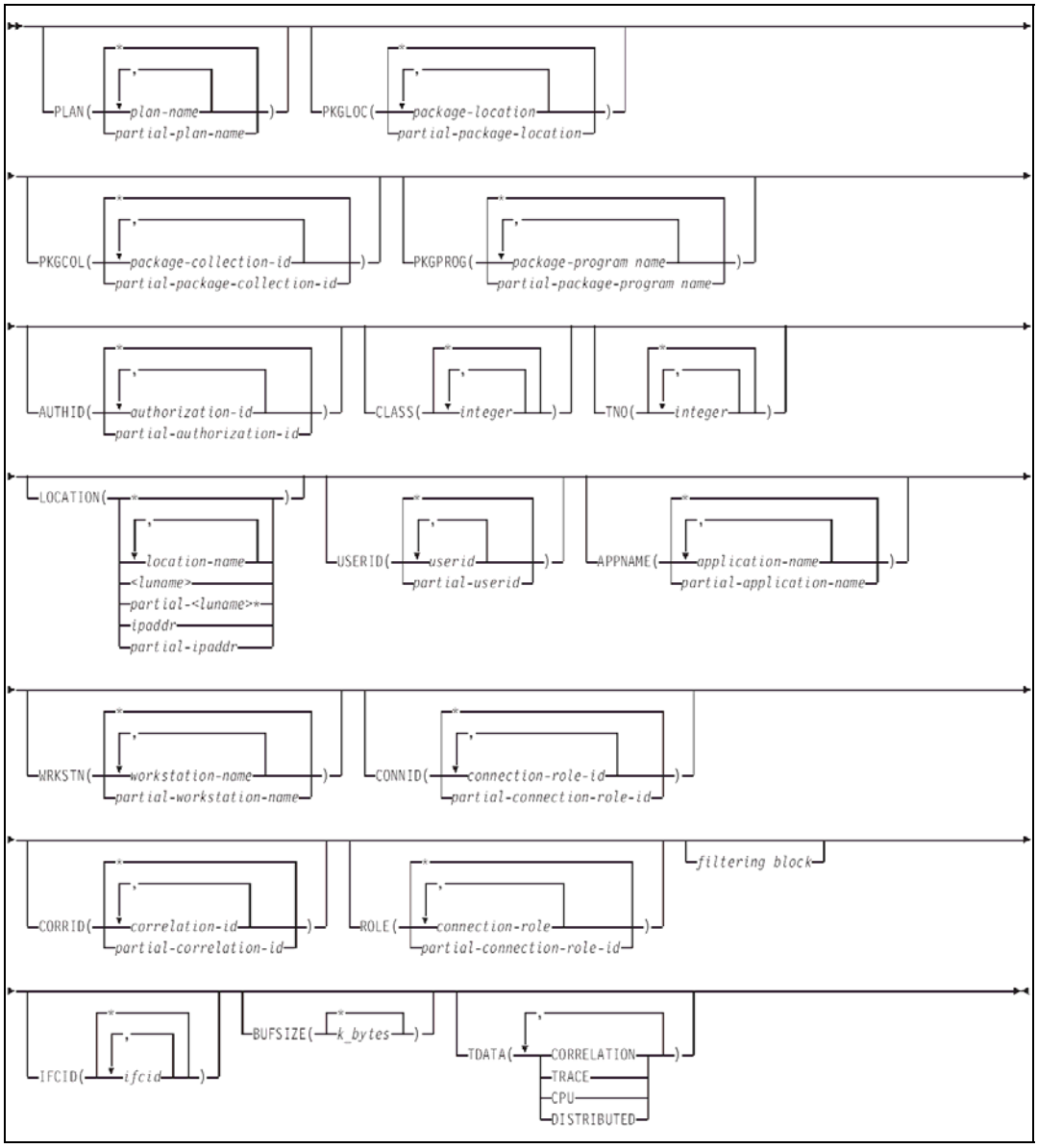


图 5.19　–constraint block

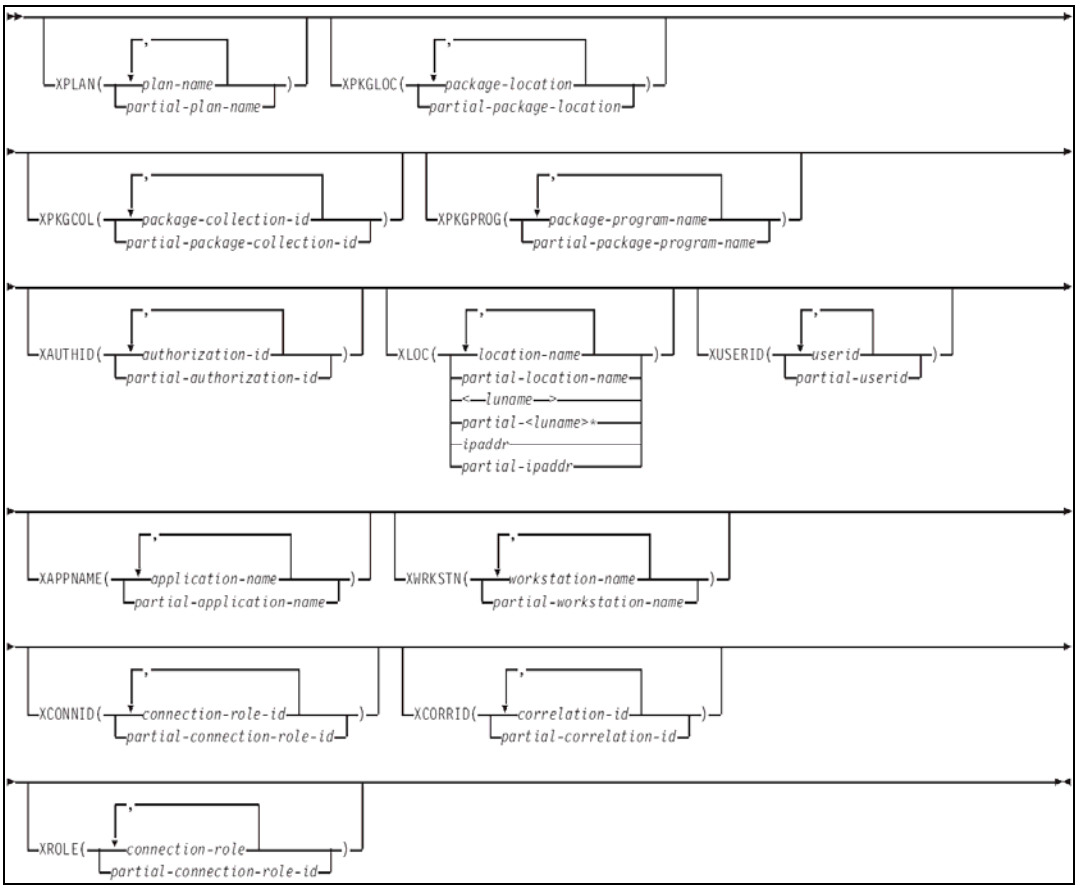


图 5.20 –filtering block（续）

5.6.2 –STOP TRACE

–STOP TRACE 停止 DB2 TRACE。缩写为：

```
-STO TRA
```

语法图如图 5.21 所示。

- 使用命令：

```
-STOP TRACE (PERFM) CLASS (1,2)
```

停止用 PERFM 选项和 CLASS (1,2)启动起来的 TRACE，而别的选项和 CLASS 启动起

来的 TRACE 则不受影响。例如用 CLASS（1）启动起来的 TRACE 就不会停止。

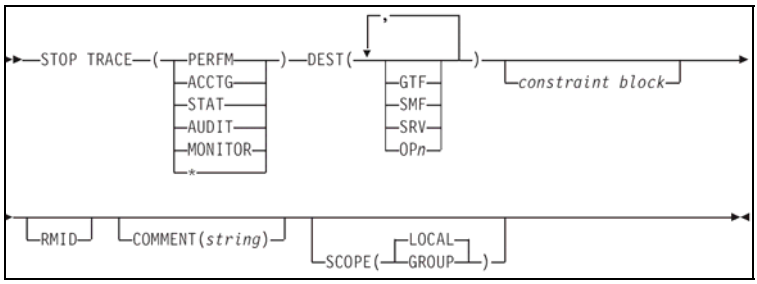


图 5.21　–STOP TRACE 语法图

- 使用命令：

```
-STOP TRACE (*)
```

停止所有的 TRACE。

5.6.3　–DISPLAY TRACE

–DISPLAY TRACE 查看当前系统起了哪些 TRACE，也可指定查看某个具体的 TRACE。语法图如图 5.22 所示。

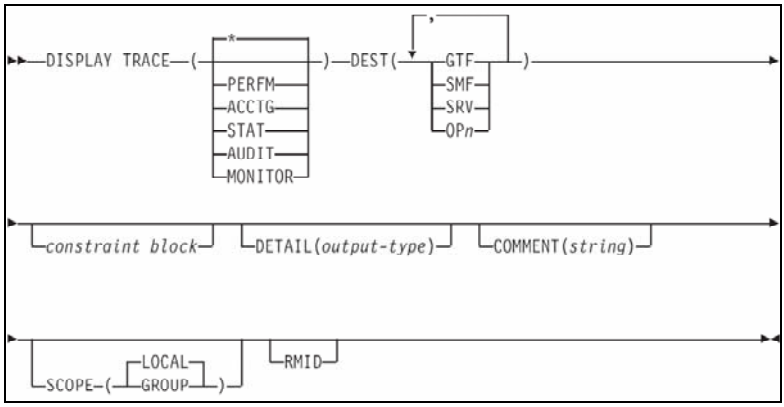


图 5.22　–DISPLAY TRACE 语法图

- 使用命令：

```
-PB11 DIS TRACE(*)
```

显示所有的 TRACE，输出如下：

```
--PB11 DIS TRACE(*)
 DSNW127I  -PB11 CURRENT TRACE ACTIVITY IS -
 TNO TYPE   CLASS         DEST QUAL
 01  STAT   01,03,04,05, SMF  NO
 01          06
 02  ACCTG  01,02,03,07, SMF  NO
 02          08
 03  MON    30           OP1  NO
 04  STAT   04           OP1  NO
 05  STAT   03           OP1  NO
 06  PERFM  30           OP1  NO
 07  ACCTG  01,02,03,07  OP2  NO
 08  MON    01           OP2  NO
 09  PERFM  31           OP2  NO
 10  MON    01           SMF  NO
 *********END OF DISPLAY TRACE SUMMARY DATA*********
 DSN9022I  -PB11 DSNWVCM1 '-DIS TRACE' NORMAL COMPLETION
```

- 使用命令：

```
-PB11 DIS TRACE=p
```

显示活动的 PERFORMANCE TRACE，输出如下：

```
--PB11 DIS TRACE=p
 DSNW127I  -PB11 CURRENT TRACE ACTIVITY IS -
 TNO TYPE   CLASS         DEST QUAL
 06  PERFM  30           OP1  NO
 09  PERFM  31           OP2  NO
 *********END OF DISPLAY TRACE SUMMARY DATA*********
 DSN9022I  -PB11 DSNWVCM1 '-DIS TRACE' NORMAL COMPLETION
```

5.7 PROCEDURE 相关命令

5.7.1 –DISPLAY PROCEDURE

–DISPLAY PROCEDURE 显示存储过程的相关统计信息，缩写为：

```
-DIS PROC
```

语法图如图 5.23 所示。

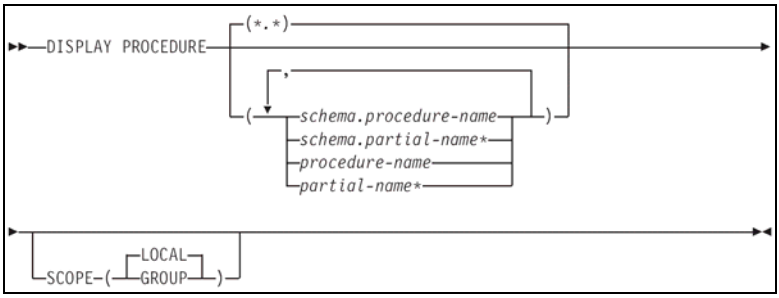


图 5.23 –DISPLAY PROCEDURE 语法图

参数说明如下。

1．SCOPE (LOCAL)

只显示当前 MEMBER 上的存储过程的信息。

2．SCOPE (GROUP)

显示所有 MEMBER 上的存储过程的信息。

- 使用命令：

```
-DIS PROC (NADBA.ASP0001) SCOPE(GROOUP)
```

显示存储过程 NADBA.ASP0001 的信息。

- 使用命令：

```
-DIS PROC (ASP0001) SCOPE(GROOUP)
```

显示存储过程 SYSPROC.ASP0001 的信息。

- 使用命令：

```
PB11 DIS PROCEDURE(*)
```

信息显示如下：

```
--PB11 DIS PROCEDURE(*)
DSNX940I  -PB11 DSNX9DIS DISPLAY PROCEDURE REPORT FOLLOWS -

-------  SCHEMA=SYSPROC
PROCEDURE       STATUS ACTIVE QUED MAXQ TIMEOUT FAIL WLM_ENV
DSNWZP
                STARTED     0    0    1       0    0 PB01WLM4
DSNX9DIS DISPLAY PROCEDURE REPORT COMPLETE
DSN9022I  -PB11 DSNX9COM '-DISPLAY PROC' NORMAL COMPLETION
```

5.7.2　–START PROCEDURE

–START PROCEDURE 命令用来启动被停掉的存储过程及刷新在 CACHE 中的存储过程。当新定义一个存储过程的时候不需要用这个命令启动它，在第一次调用该存储过程时，DB2 会自动启动激活存储过程。命令缩写为：

```
-STA PROC
```

语法图如图 5.24 所示。

参数说明如下。

1．PROCEDURE

如果没有指明 SCHEMA，则默认的 SCHEMA 是 SYSPROC。

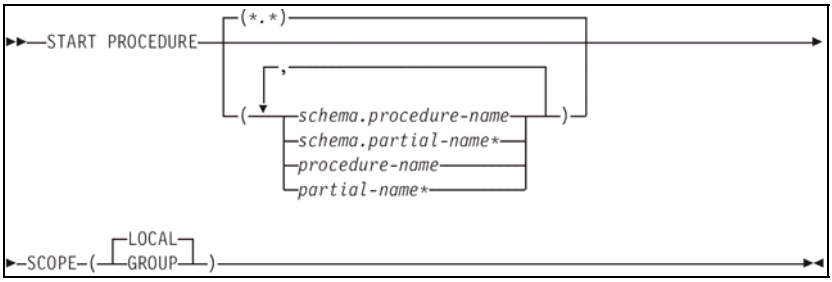


图 5.24　–START PROCEDURE 语法图

2．SCOPE

说明这个 procedure 的范围。

LOCAL　只启动当前 MEBER 上的指定 PROCEDURE。

GROUP　启动所有 MEMBER 上的指定 PROCEDURE。

- 使用命令：

```
-STA PROC(*.*)
```

或者

```
-STA PROC
```

启动所有 SCHEMA 下的存储过程。

- 使用命令：

```
-STA PROC(NGDBA.SP1) SCOPE(LOCAL)
```

只在当前 MEMBER 中启动存储过程 NGDBA.SP1。此外在存储过程名字中也可以使用通配符。

5.7.3　–STOP PROCEDURE

–STOP PROCEDURE 停掉 DB2 的一个或多个存储过程。缩写为：

```
-STO PROC
```

语法图如图 5.25 所示。

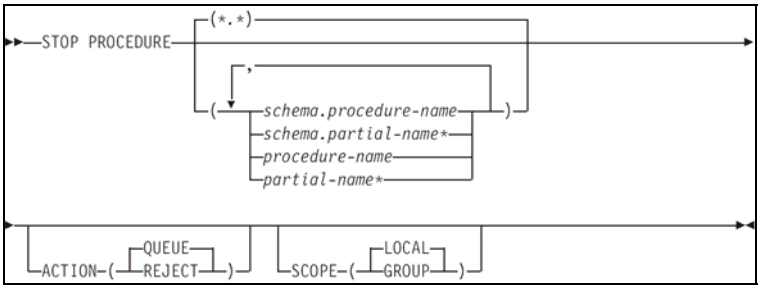


图 5.25　–STOP PROCEDURE 语法图

参数说明如下。

1．PROCEDURE-NAME和PARTIAL-NAME*

如果没有指定 PROCEDURE 的 SCHEMA，那么默认 SCHEMA 就是 SYSPROC。

2．ACTION

指 PROCEDURE 被停止后，对这个 PROCEDURE 的访问请求如何处理。

QUEUE：访问请求进行排队，直到等待超时了或者存储过程重新启动。

REJECT：所有的访问请求被拒绝。

3．SCOPE

指这个命令的作用范围。

LOCAL：只停掉当前 MEMBER 上的指定存储过程。

GROUP：停掉整个 GROUP 上的指定存储过程。

5.8　DDF 相关命令

5.8.1　–START DDF

–START DDF 启动 DDF，缩写为：

```
-STA DDF
```

语法图如图 5.26 所示。

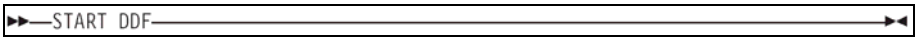


图 5.26　–START DDF 语法图

5.8.2 –STOP DDF

–STOP DDF 停止 DDF，缩写为：

```
–STP DDF
```

语法图如图 5.27 所示。

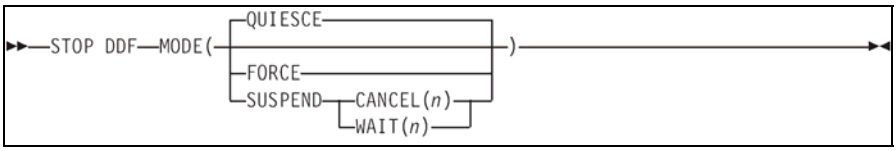


图 5.27 –STOP DDF 语法图

参数说明如下。

1．MODE

指正在执行的 distributed threads 如何处理，是否允许其正常结束。

QUIESCE：允许所有活动的 distributed threads 正常结束，结束所有的 inactive distributed threads。

FORCE：终止所有正在执行的 distributed threads。

SUSPEND：用下面三种方式挂起所有的 DDF 线程。

- 保持 inactive DDF 线程为 inactive 状态，直到发出 START DDF 命令。
- 终止所有的 DDF pool 线程。
- 阻止 inbound DDF 启动。

2．CANCEL (n)

指在 *n* 秒内没有完成上面的 DDF 挂起过程，就 CANCEL 掉所有的活动 DDF 线程

3．WAIT (n)

指在 *n* 秒内没有完成上面的 DDF 挂起过程，就继续 DDF 的处理。

- 使用命令：

```
-PB11 STP DDF FORCE
```

终止所有的活的线程停止 PB11 上的 DDF。

- 使用命令：

```
-PB11 STP DDF QUIESCE
```

允许活动的线程正常结束后再停止 DDF。

5.8.3　–DISPLAY DDF

- 使用命令：

```
-PB11 DIS DDF
```

显示信息如下：

```
--PB11 DIS ddf
 DSNL080I  -PB11 DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
 DSNL081I STATUS=STARTD
 DSNL082I LOCATION           LUNAME             GENERICLU
 DSNL083I DSNPB01            VTAM1.PB11LU       VTAM1.PB01GRP
 DSNL084I IPADDR          TCPPORT RESPORT
 DSNL085I 83.16.49.31     4461    4481
 DSNL086I SQL     DOMAIN=dsnpb01.plexpt1.icbc.com
 DSNL086I RESYNC DOMAIN=pb11lu.dsnpb01.plexpt1.icbc.com
 DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

- 使用命令：

```
-PB11 DIS DDF DETAIL
```

显示 DDF 的信息及 DDF 的统计和配置信息。显示信息如下：

```
--PB11 DIS DDF DETAIL
 DSNL080I  -PB11 DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
 DSNL081I STATUS=STARTD
 DSNL082I LOCATION           LUNAME             GENERICLU
 DSNL083I DSNPB01            VTAM1.PB11LU       VTAM1.PB01GRP
 DSNL084I IPADDR          TCPPORT RESPORT
 DSNL085I 83.16.49.31     4461    4481
 DSNL086I SQL     DOMAIN=dsnpb01.plexpt1.icbc.com
 DSNL086I RESYNC DOMAIN=pb11lu.dsnpb01.plexpt1.icbc.com
 DSNL090I DT=A  CONDBAT=     64 MDBAT=    40
 DSNL092I ADBAT=     2 QUEDBAT=       0 INADBAT=       0 CONQUED=       0
 DSNL093I DSCDBAT=     0 INACONN=       0
 DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

5.9　LOG 相关命令

5.9.1　–ARCHIVE LOG

–ARCHIVE LOG 关闭当前的 ACTIVE LOG，并将其归档，启用下一个 ACTIVE LOG。语法图如图 5.28 所示。

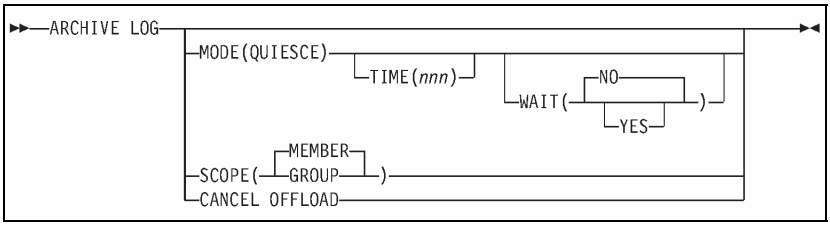


图 5.28 –ARCHIVE LOG 语法图

参数说明如下。

1．MODE(QUIESCE)

终止所有新的对数据库的更新活动，试图通过提交或回滚动作让现有的线程达到一个统一的一致点，然后关闭现用的 ACTIVE LOG 启用下一个 LOG。

2．TIME(nnn)

设置一个最大时间，以秒计算，这个时间内试图达到全系统的 QUIESCE，如果这个时间内没有达到全局的 QUIESCE，那么 ARCHIVE LOG 命令就会失败。如果没有设置这个值，那么默认值是在安装 DB2 的时候在 DSNTIPA 面板的 QUIESCE PERIOD 的值。

3．WAIT

指定 DB2 是否等待 QUIESCE 过程结束才把控制返回到调用该命令的终端控制台或程序。

NO：指当 QUIESCE 过程一开始就必须把控制返回给调用该命令的终端控制台或程序。如果用这个参数，对用户来说，QUIESCE 过程就是异步的。控制被返回给控制台后用户就可以发出其他的 DB2 命令，但是直到 ARCHIVE LOG 命令结束后这些命令才会被命令处理器处理。

YES：指要等 QUIESCE 过程结束后才把控制返回给调用该命令的终端控制台或程序。如果用这个参数，对用户来说，QUIESCE 过程就是同步的。

4．SCOPE

指定命令的作用范围，只有在 Data Sharing 环境中这个参数才有效，如果不是 Data Sharing 环境，这个参数就会被忽略。

MEMBER：指这个命令只作用于当前 MEMBER。如果这个 MEMBER 正在 ARCHIVE，

这个命令就会失败。

GROUP：指这个命令作用于整个 group。如果 group 中的任何一个 MEMBER 正在 ARCHIVE，这个命令就会失败。

使用 ARCHIVE LOG 命令的一些注意点如下。

- 当 DB2 正在处理 STOP DB2 MODE(QUIESCE)或者 STOP DB2 MODE(FORCE)的时候，不能执行带 MODE(QUIESCE)参数的 ARCHIVE LOG 命令，如果在这种环境中发出了带 MODE(QUIESCE)参数的 ARCHIVE LOG 命令就会返回错误代码 DSNJ315I 或者 DSNJ316I。
- 当 DB2 正在处理 STOP DB2 MODE(QUIESCE)的时候，允许执行不带 MODE(QUIESCE)参数的 ARCHIVE LOG 命令，当 DB2 正在处理 STOP DB2 MODE(FORCE)的时候，不允许执行 ARCHIVE LOG，如果在处理 STOP DB2 MODE(FORCE)的时候，发出 ARCHIVE LOG 命令就会返回错误代码 DSNJ315I。
- 当 DISPLAY THREAD 命令返回消息 DSNV400I 的时候，说明一个 ARCHIVE LOG MODE(QUIESCE)正在执行。
- 使用命令：

```
-DB1G ARCHIVE LOG MODE(QUIESCE) TIME(600)
```

设定 QUIESCE 阶段的时间为 10 分钟，在这个阶段所有的 UPDATE 都被停止掉，如果 10 分钟的 QUIESCE 阶段没有达到 DB1G 这个 MEMBER 的全局 QUIESCE，命令就会失败。

- 使用命令：

```
-DB2G ARCHIVE LOG
```

截断 DB2G 的 ACTIVE LOG 并且切换到下一个可用的 LOG，初始化一个异步作业来 OFFLOAD 这个被截断的 LOG。

5.9.2　–DISPLAY LOG

–DISPLAY LOG 查看 ACTIVE LOG 状态、LOGLOAD 的设置及 OFFLOAD 的状态。

- 使用命令：

```
–DISPLAY LOG
```

输出信息格式如下：

```
--PB11 DIS log
 DSNJ370I   -PB11 DSNJC00A LOG DISPLAY
 CURRENT COPY1 LOG = DSNPB01.PB11.LOGCOPY1.DS06 IS 74% FULL
 CURRENT COPY2 LOG = DSNPB01.PB11.LOGCOPY2.DS06 IS 74% FULL
           H/W RBA = 0B4C62459656
           H/O RBA = 0B4C41F46FFF
           FULL LOGS TO OFFLOAD = 0 OF 32
           OFFLOAD TASK IS (AVAILABLE)
 DSNJ371I   -PB11 DB2 RESTARTED 19:49:49 NOV 18, 2009
           RESTART RBA 0B40E22D1000
           CHECKPOINT FREQUENCY 5 MINUTES
           LAST SYSTEM CHECKPOINT TAKEN 14:33:38 NOV 24, 2009
 DSN9022I   -PB11 DSNJC001 '-DIS LOG' NORMAL COMPLETION
```

5.10 THREAD 相关命令

5.10.1 –CANCEL THREAD

–CANCEL THREAD 取消某 DB2 线程。

语法图如图 5.29 所示。

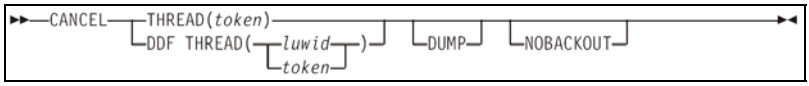


图 5.29 –CANCEL THREAD 语法图

参数说明如下。

THREAD（token）

token 作为每一个线程的唯一的表示，是一个一位到六位的数字。可以用 DISPLAY THREAD 命令来查看线程对应的 token，token 为 0 的线程不能 CANCEL 掉。

- 使用命令：

```
-CANCEL THREAD (45162)
```

CANCEL 掉 token 号为 45162 的线程。

5.10.2 –DISPLAY THREAD

–DISPLAY THREAD 查看 DB2 THREAD 状态。

语法图如图 5.30 所示。

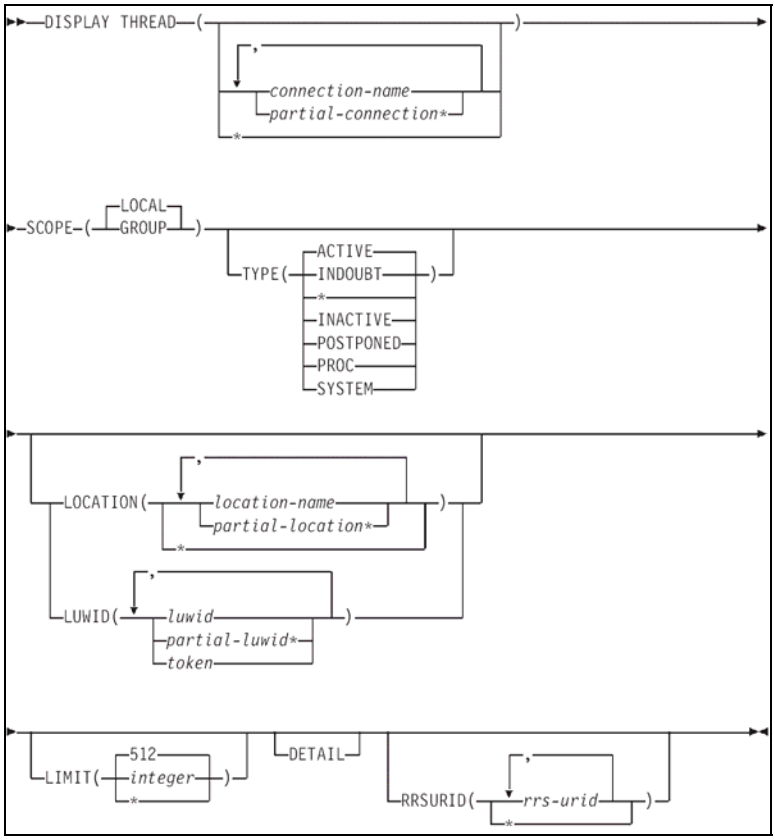


图 5.30　–DISPLAY THREAD 语法图

参数说明如下。

TYPE：显示的线程类型。

ACTIVE：只显示 ACTIVE 的线程，缩写为 A。

INDOUBT：只显示 INDOUBT 线程，缩写为 I。

INACTIVE：只显示 INACTIVE 线程，缩写为 INA。

POSTPONED：显示 BACK-OUT PROCESSING 被推迟线程信息，缩写为 P。

PROC：只显示那些运行在存储过程和用户自定义函数里面的线程信息。

SYSTEM：显示部分 SYSTEM AGENT 线程信息，缩写为 SYS。

- 使用命令：

```
-DISPLAY THREAD(*)
```

查看所有的 THREAD，每一个 THREAD 都对应有一个 TOKEN，可以用这个 TOKEN 来 CANCEL 这个 THREAD。

- 使用命令：

```
-DISPLAY THREAD(*)  LOCATION(*)  DETAIL
```

显示所有线程的详细信息，包括 active，inactive 和 indoubt 三类状态。

- 使用命令：

```
-DISPLAY THREAD(*) TYPE(INDOUBT)
```

显示所有 INDOUBT 类型的线程信息。

- 使用命令：

```
-PB11 DIS thread(*)
```

信息显示如下：

```
--PB11 DIS thread(*)
 DSNV401I  -PB11 DISPLAY THREAD REPORT FOLLOWS -
 DSNV402I  -PB11 ACTIVE THREADS -
 NAME     ST A    REQ ID           AUTHID   PLAN     ASID TOKEN
 SERVER   RA *     40 db2jcc_appli BJB0051  DISTSERV 00E5  3845
  V437-WORKSTATION=kfzxzhuhkp, USERID=BJB0051,
       APPLICATION NAME=db2jcc_application
  V445-L30ABF50.G769.C522882E8451=3845 ACCESSING DATA FOR 83.10.191.80
 SERVER   RA *      0 db2jcc_appli BJB0011  DISTSERV 00E5 166402
  V437-WORKSTATION=dccbtianjs1, USERID=BJB0011,
       APPLICATION NAME=db2jcc_application
  V445-L30ABF94.G5F2.C522807F9C22=166402 ACCESSING DATA FOR
       83.10.191.148
 RRSAF     T     5188               CANUSR   KO2PLAN  00F5     1
 RRSAF     T     5742               CANUSR   KO2PLAN  00F5     2
 RRSAF     T    16744               CANUSR            00F5     3
 RRSAF     T    15375 OMEGAMON      CANUSR   KO2PLAN  00F5    40
 CI11PE11 N        3                STCUSER           0149     0
 CI11PE12 N        3                STCUSER           0188     0
 CI11PA11 N        3                STCUSER           0106     0
 CI11PA11 ND   12556 ENTR74030001    NGDBA             0106     0
```

5.11 IRLM 相关命令

5.11.1 –START irlmproc

–START irlmproc 启动在安装 DB2 时指定的 IRLMPROC。缩写为：

```
S irlmproc
```

语法图如图 5.31 所示。

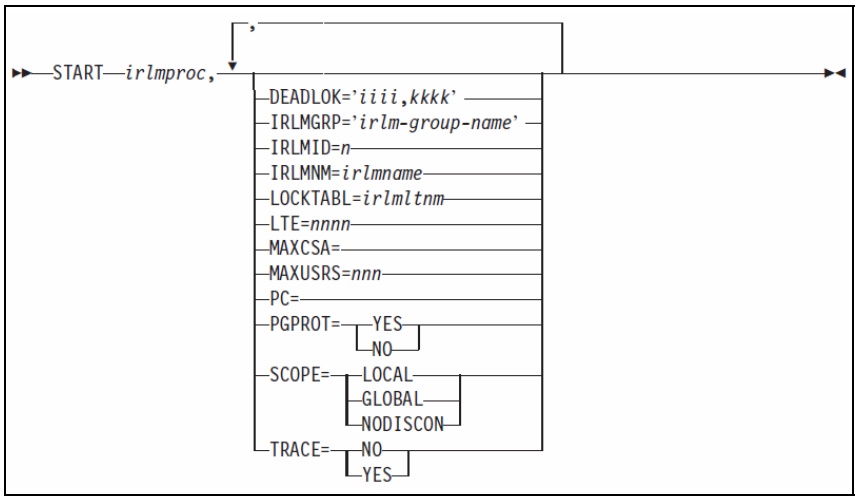


图 5.31　–START irlmproc 语法图

参数说明如下。

1．IRLMPROC

要启动的 IRLMPROC 名字，例如：PB11IRLM。

下面的参数都是可选参数。

- DEADLOK=’ iiii,kkkk ’

iiii 接受的范围是 1～9999，指定 local deadlock detection 的时间间隔，如果赋值大于 5，IRLM 就默认用 5。

kkkk 接受的范围是 1～9999，指定在 global deadlock detection 开始前允许的 local deadlock detection cycles 的次数。如果赋值大于 1，IRLM 就默认用 1。

- RLMGRP=’ irlm-group-name ’

在 data-sharing 环境中，这个 IRLM 属于的 XCF GROUP 名字。在 non-data-sharing 中（SCOPE=LOCAL）这个参数被忽略。

- IRLMID= n

当前 MEMBER 的 IRLM 的 ID，用来区别同一个 GROUP 中各个 IRLM。这里 N 可以是 1～255 的数字，也可以是可打印字符（要用单引号引起来，例如 IRLMID=‘D’）。当是可打印字符时，IRLMID 取值为这个字符的 EBCDIC 值。IRLMNM= irlmname 为当前子系统的 IRLM 名字，4 位的字符。

- LOCKTABL= irlmltnm

指定这个 GROUP 用到的 lock table，如果使用这个参数，需要在 IMS 环境中。在 non-data-sharing 中。在（SCOPE=LOCAL）这个参数被忽略。

- LTE= nnnn

指定在 CF LOCK structure 需要的 lock table entries 数量，这个数值可以是空值，可以是零，2 的整数平方，最高可以是 1024。

- MAXCSA=

这个参数是预留参数，现在的版本中没有使用。

- MAXUSRS= nnn

初始化 GROUP 中最多可以有多少个 MEMBER，这个值决定了在 LOCK TABLE 中的每一条 LOCK ENTRY 是多少字节，图 5.32 显示了不同的值对 LOCK ENTRY 的影响。

MAXUSRS	Initial size of lock entry
7 or less	2 bytes
≥ 8 and < 24	4 bytes
≥ 24 and < 33	8 bytes

图 5.32　不同的值对 LOCK ENTRY 的影响

2．PC

这个参数是预留参数，现在的版本中没有使用。

3．PGPROT

可以取值为 YES 和 NO，指定 IRLM LOAD-MODEL 是否放在 page-protected storage 上。

4．SCOPE

可以取值为 LOCAL （在 non-data-sharing 环境中），GLOBAL(在 data-sharing 环境中)和 NODISCON。

5．TRACE

指定在 wrap-around IRLM buffers 中是否捕捉 TRACE。

- YES：在 wrap-around IRLM buffers 中捕捉 TRACE。
- NO：不捕捉 TRACE，直到 TRACE CT 命令发出。

5.11.2　–STOP irlmproc

–STOP irlmproc 停止指定的 IRLMPROC。缩写为：

```
P IRLMPROC
```

语法图如图 5.33 所示。

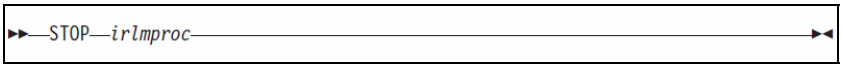


图 5.33　–STOP irlmproc 语法图

如果 STOP 命令没有使 IRLM 正常停止，就可以用 MODIFY irlmproc,ABEND 命令来 TERM 掉 IRLM。

5.11.3　–TRACE CT

–TRACE CT 命令用来启动、停止和修改在 IRLM 上的 TRACE。

语法图如图 5.34 所示。

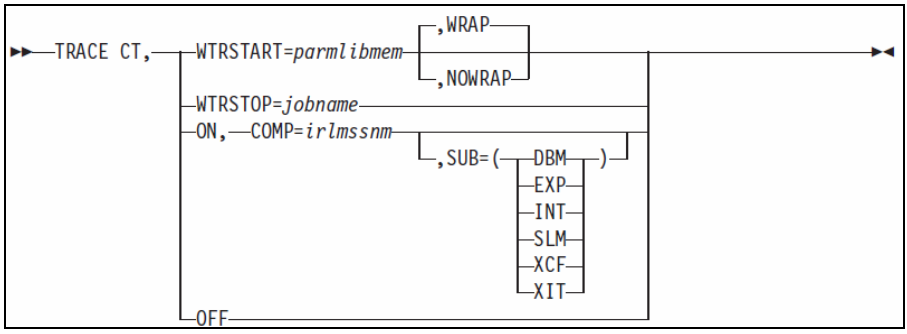


图 5.34　–TRACE CT 语法图

参数说明如下。

1．ON

说明打开一个 TRACE ,COMP=IRLMNM 指定 IRLM 子系统的名字。

2．SUB=

指定 TRACE 的子类型。

- DBM：和指定的 DBMS 的交互。
- EXP：任何例外情况。
- INT：正常锁活动之外的 member 和 group 事件。
- SLM：和 z/OS locking 组件的交互。
- XCF：和 z/OS cross-system coupling services 之间所有的交互。
- XIT：和 z/OS locking 组件的异步交互。

3．OFF

说明是关掉 TRACE。

5.11.4 –MODIFY irlmproc, ABEND

–MODIFY irlmproc, ABEND 用来 TERM IRLM，缩写为：

```
F irlmproc,ABEND
```

语法图如图 5.35 所示。

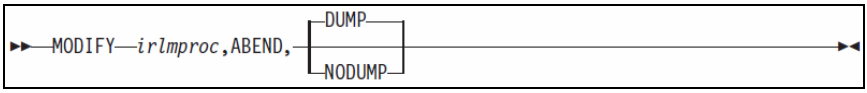


图 5.35 –MODIFY irlmproc, ABEND 语法图

参数说明如下。

- **DUMP**：TERM 掉指定 IRLM，在 SYS1.DUMPxx 中有系统 DUMP 信息。
- **NODUMP**：强制 TERM 指定的 IRLM 并且没有做系统 DUMP。

使用命令：

```
F PB11IRLM ABEND
```

强制 TERM 掉 PB11IRLM，并且产生 DUMP 信息。

5.11.5 –MODIFY irlmproc, DIAG

–MODIFY irlmproc, DIAG 指定这个 IRLM 子系统的初始化 diagnostic DUMP，也就是在什么情况发生时就做一个 DUMP。缩写为：

```
F irlmproc,DIAG
```

语法图如图 5.36 所示。

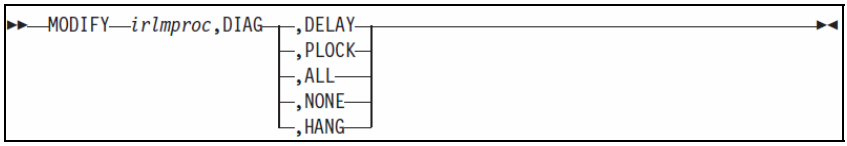


图 5.36　–MODIFY irlmproc, DIAG 语法图

参数说明如下。

1．DELAY

当传播到 CF 的子锁持续时间超过了 45 秒时，就会在 SYS1.DUMP*xx* 数据集中产生 diagnostic DUMP 信息。

2．PLOCK

当 IRLM 检测到 P-lock negotiation 超过了 2 分钟时，就会进行 diagnostic DUMP。

3．ALL

上述两种情况都产生 diagnostic DUMP 信息。

4．NONE

无论什么情况都不产生 diagnostic DUMP 信息。

5．HANG

当 IRLM 检测到 DEADLOCK 或者 TIMEOUT 可能出现时，IRLM 就会收集 IRLM SYSPLEX 中的 DUMP 信息。

- 使用命令：

```
MODIFY PB11IRLM,DIAG,DELAY
```

每当 propagation of child locks 持续时间超过了 45 秒，就会进行一次 DUMP。

5.11.6　–MODIFY irlmproc, PURGE

–MODIFY irlmproc, PURGE 释放某个 inactive 的 DB2 子系统在 IRLM 中的所有锁，这些锁可能是由于 DB2 IRLM 或系统报错等原因造成。

语法图如图 5.37 所示。

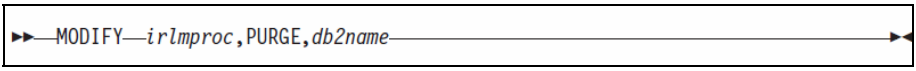


图 5.37　–MODIFY irlmproc,PURGE 语法图

参数说明如下。

1．irlmproc

处理这条命令的 IRLMPROC。

2．db2name

要释放锁的 DB2 子系统，一般是用 STATUS 命令显示的 inactive DB2 子系统。

- 使用命令：

```
F PB11IRLM,STATUS,ALLD
```

PB11 是一个活动的 DB2 子系统，它的 irlmproc 是 PB11IRLM，上面这条命令就会显示出这个 Data Sharing SYSPLEX 中所有的 active 和 inactive 的子系统。如果 PB21 是 inactive 的，那么使用命令：

```
F PB11IRLM,PURGE,PB21
```

命令执行后，会返回如下信息：

```
DXR109I IR2B002 PURGE COMMAND COMPLETED FOR PB21
```

5.11.7　–MODIFY irlmproc, SET

–MODIFY irlmproc, SET 命令用来动态设置 IRLM 不同的参数。可以设置的参数如下：

- IRLM 允许的最大 private storage。
- IRLM 允许的 trace buffers 数目。
- 下次连接 XCF LOCK 时的 LOCK LTE entries 数目。
- 对某个 DB2 系统设置 TIMEOUT 值。
- 设置 local deadlock frequency。

语法图如图 5.38 所示。

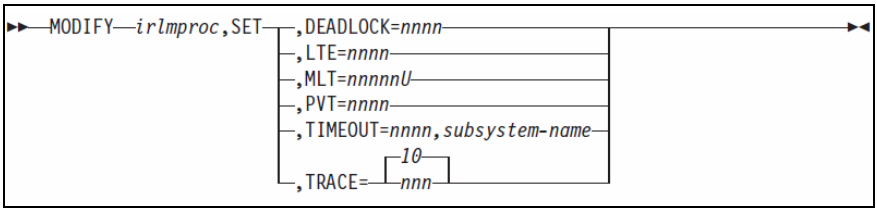


图 5.38　–MODIFY irlmproc,SET 语法图

参数说明如下。

- **DEADLOCK** 和 **LTE** 可以参照命令 START irlmproc。
- **MLT=** *nnnnnU* ：设置在 2GB 线以上的 private storage 的最大上限，nnnnn 取值范围是 0～99 999，U 是容量单位（M，G，T，P）。这个值必须大于当前使用的 private storage 和 2GB 线。MLT 这个值设置是临时的，只在当前 IRLM instance 运行期间有效，下次重新启动后就会用启动时的参数值，所以要永久修改这个值必须修改 IRLM 启动 PROC 作业的 MEMLIMIT JCL EXEC 值。
- **PVT=** *nnnn* ：设置在 2GB 线以下的 private storage 的最大上限，nnnnn 取值范围是 0～1800，U 是容量单位（M，G）。如果设置的最大值小于当前 IRLM 已经使用的 private storage，这个命令不被执行并且返回消息 DXR106E。
- **TIMEOUT=** *nnnn,subsystem-name*：对指定的 DB2 子系统设置 TIMEOUT 值，nnnn 取值为 1～3600，*subsystem-name* 是 DB2 子系统的名字。
- **TRACE=** *nnn*：设置每个 TRACE 的 64KB 的 trace buffers 最大数目限制，nnn 取值为 10～255。

5.11.8　–MODIFY irlmproc, STATUS

–MODIFY irlmproc, STATUS 显示连接到指定 IRLMPROC 的 DB2 子系统的信息。可以显示所有连接到这个 IRLMPROC 的 DB2 子系统的信息包括：subsystem name，status，work unit，lock information，IRLM TIMEOUT 的当前值，DEADLOCK 值。

语法图如图 5.39 所示。

参数说明如下。

1．Irlmx

是 IRLM 子系统名字和 IRLM ID 的组合。其中 *Irlm* 是在 IRLM 启动 PROC 中设置的

IRLMNM 值，*x* 是在 IRLM 启动 PROC 中设置的 IRLMID 值。

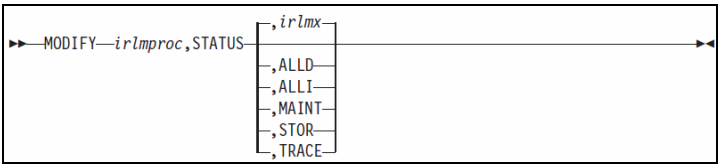


图 5.39　–MODIFY irlmproc,STATUS 语法图

2．ALLD

显示和这个 IRLMPROC 相关的 DB2 子系统的名字和状态。如果 DB2 状态是 DOWN，但是它拿着 retained 锁，那么它也会显示。

3．ALLI

显示和这个 IRLMPROC 相关的 DB2 子系统的名字、ID、状态和 service level。如果 DB2 状态是 DOWN，但是它拿着 retained 锁，那么它也会显示。

4．TRACE

显示 IRLM 的 TRACE 类型的相关信息，每个 TRACE 用了多少 trace buffers 及 trace external writer 是否 ACTIVE。

- 使用命令：

```
-modify pb11irlm,status,alld
```

输出信息如下：

```
-modify pb11irlm,status,alld
 DXR102I PJ11001 STATUS
 SUBSYSTEMS IDENTIFIED
         NAME      STATUS     RET_LKS    IRLMID   IRLM_NAME    IRLM_LEVL
         PB11      UP               0     001       PJ11          2.025
         PB21      UP               0     002       PJ21          2.025
         PB31      UP               0     003       PJ31          2.025
         PB41      UP               0     004       PJ41          2.025
     DXR102I End of display
```

- 使用命令：

```
-modify pb11irlm,status,alli
```

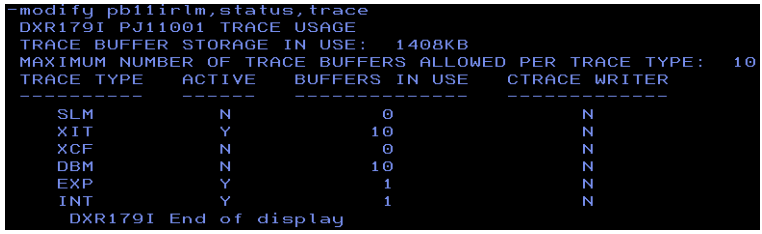
输出信息如下：

```
-modify pb11irlm,status,alli
 DXR103I PJ11001 STATUS
         IRLMS PARTICIPATING IN DATA SHARING GROUP FUNCTION LEVEL=2.025
         IRLM_NAME  IRLMID STATUS LEVEL SERVICE MIN_LEVEL MIN_SERVICE
           PJ11       001      UP    2.025 PK05211    1.022      PQ52360
           PJ21       002      UP    2.025 PK05211    1.022      PQ52360
           PJ31       003      UP    2.025 PK05211    1.022      PQ52360
           PJ41*      004      UP    2.025 PK05211    1.022      PQ52360
     DXR103I End of display
```

- 使用命令：

```
-modify pb11irlm,status,trace
```

输出信息如下：



由于篇幅有限，本章只列出主要的 DB2 命令和常用的使用方法举例，命令的详细语法和使用限制可以参考《DB2 UDB for z/OS V8 Command Reference》。

5.12　课后习题

一、选择题：

1．要在 PT11 上启动 pb11 这个子系统，应该用下面的哪条命令？（　　）

A．/pt11 –pb11 START DB2

B．/pt11 –START DB2

C．/pt11 –pb11 DB2 START

D．/pt11 –pb11 START

2．如果表空间 nasedb.spname 状态是 RECOVER-pending 的状态，如果要用重启表空间的命令来重置表状态，应该用下面哪个命令？（　　）

A．-STA DB(NASEDB) SP(SPANME) ACCESS（RW）

B．-STA DB(NASEDB) SP(SPANME) ACCESS（UT）

C．-STA DB(NASEDB) SP(SPANME) ACCESS（FORCE）

D．-STA DB(NASEDB) SP(SPANME) ACCESS（RO）

3．在 DB2 中要杀掉一个 Utility-id 是 123 的 Utility，用下面哪个命令？（　　）

A．-CANCEL UTIL（123）

B．-STOP UTIL (123)

C．-PURGE UTIL （123）

D．-TERM UTIL (123)

4．查看有哪些交易或访问连接到了当前 DB2，可以用下面哪个命令？（　　）

A．DIS THREAD

B．DIS LOG

C．DIS TRACE

D．DIS LOCATION

二、填空题

1．在 BIND PACKAGE 和 BIND PLAN 的时候，使用参数（　　）就可以保留用户对这个 PLAN 以前的执行权限。

2．如果想看当前 LOG DATASET 的使用情况，可以用命令（　　）。

3．在 LOAD 的时候如果作业断掉，这个表空间一直处于 UT 状态，可以用命令（　　）查看是否有不活动的 Utility 锁住这个表空间。

三、简答题

1．表空间有哪几种 PENDING 状态？

2．索引有哪几种 PENDING 状态？

3．当用 DISPLAY DDF 命令查看 DB2 的 IP 和端口的时候，如果发现没有获取到 IP，可能的原因是什么？如何处理？

第 6 章　DB2 系统维护概述

本章为 DB2 系统维护方法介绍，主要包括系统备份和恢复、日常监控和健康检查、例行重组和性能分析调优等内容，旨在帮忙读者对 DB2 维护工作有较为全面的了解。

6.1　DB2 备份和恢复

数据是业务运行的基础，维护数据完整性和正确性是 DBA 最重要的工作职责之一，因此 DB2 系统的备份和恢复工作是日常维护工作的重中之重。本节将介绍如何建立完整的数据备份和恢复机制，详细讲述备份及恢复工具的使用及注意事项。

6.1.1　DB2 备份及恢复的原则

在进行 DB2 备份策略制定的时候，要综合备份和恢复的可行性，具体要考虑如下几点：

（1）具有可恢复性，各类备份有相应的恢复程序并可行，备份数据具有可用性。

（2）遵循“黄金原则”，对于所有关键数据至少应该有两份拷贝，每一份拷贝应该使用不同的存储介质，且存放于不同的物理地点，用于本地出现灾难后最低限度的数据恢复。

（3）保全各类备份技术资料和各次备份记录，并保证资料与实际相符。

6.1.2 如何制定最佳的备份策略

我们需要明确以下几点后，方可进行备份策略的制定。

1. 明确备份的范围

确定哪些数据需要进行备份。一般来说分为数据库系统表的备份和应用表的备份。数据库系统表空间建议每日进行完全备份，用于数据库系统发生故障时的恢复，其备份内容主要包括 DSNDB01/DSNDB06 两个 DB 中的数据。数据库应用表的备份需要根据业务及应用来决定备份的内容和备份模式（完全备份或增量备份）。

2. 明确备份的频率

确定数据备份的频率，哪些数据需要每天备份，哪些数据需要每周备份。一般来说数据库系统表的备份建议每日进行，而应用表的备份可以根据业务和应用的需求确定备份频率。

3. 明确备份方式

根据数据类型的不同，确定数据备份采用的方式，哪些可以使用 COPY Utility，哪些可以使用 UNLOAD Utility。一般来说数据库系统表使用 COPY Utility 进行备份，而应用表可以根据情况选择 COPY 或者 UNLOAD Utility 来进行备份。

4. 明确备份使用的介质

确定哪些数据的备份需要放在磁盘上，哪些数据的备份只需要保留在磁带上。一般来说对于需要快速进行恢复的备份建议保存在磁盘上，例如关键的系统表、数据库日志等，而那些对恢复时间窗口要求较少的备份，可以考虑存放在磁带上。

5. 明确备份保留的期限

需要根据应用及业务需求，制定数据库表备份的保留期限。一般来说，普通日的备份保留 1～2 个月，特殊日（月末、年终等）备份可以考虑进行永久保留。

6. 明确备份的命名规范

数据库备份文件的命名规范需要统一化。

7. 明确恢复的方式

恢复的方式一般由备份方式决定，如果使用 COPY 进行的备份，可以使用 RECOVER

或者 DSN1COPY 进行恢复；如果使用 UNLOAD 方式进行的备份，可以使用 RELOAD 方式进行恢复。

6.1.3　DB2 备份常用工具及使用方法

在 DB2 日常维护中，对数据库表进行备份可以采用 COPY 和 UNLOAD 两类 Utility，COPY 是基于页面的复制，UNLOAD 是将数据库表中的记录下载生成顺序文件。下文将详细介绍这两种数据库表备份的方法。

6.1.3.1　DB2 COPY Utility

COPY 也称 IMAGECOPY，是对表空间（TABLESPACE）或者索引空间（INDEXSPACE）的操作，COPY 是基于页面级拷贝。分为 FULL IMAGE COPY 和 INCREAMENTAL IMAGE COPY 两种方式。FULL IMAGE COPY 是 COPY DB2 表中所有的 PAGE，而 INCREAMENTAL IMAGE COPY 是备份从上次 FULL IMAGE COPY 到现在之间发生变化的 PAGE，一般常用 FULL COPY 对数据库表进行备份。COPY Utility 的相关参数，请参考 Utility 的相关章节。

另外，由于在使用 RECOVER Utility 进行数据库表恢复的时候，需要数据库的备份及数据库日志配合完成，因此数据库备份的保留期限需要和数据库日志的保留期限一致。例如数据库的备份如果保留两周，那么对应数据库的 ARCHIVE LOG 也需要保留两周。

在对数据库进行 COPY 操作的时候，用户需要拥有 IMAGECOPY 权限，或者拥有 DBADM、DBCRTL、DBAMINT 等管理权限。

在使用 COPY Utility 对数据库表进行备份后，在 SYSIBM.SYSCOPY 表中会增加相应的记录，此记录包含了此备份的相关信息，例如备份的表空间名称、此表空间对应的 DB 名称及 DB2 GROUP 名称、备份日期、备份介质、备份生成的文件名称、备份作业名称等。当备份作业也就是 COPY Utility 正确执行后会将上述信息插入到 SYSIBM.SYSCOPY 表中。随着备份执行次数的增多，SYSCOPY 表中的记录会持续增长，对应表空间也不断增长，一方面影响恢复的效率，也对维护带来一定的风险。因此需要配合备份的到期日进行清理工作，通常使用 MODIFY Utility 对过期的备份信息进行定期清理。

6.1.3.2　DB2 UNLOAD Utility

上一节介绍了基于表空间的备份方法 COPY，本节介绍另外一种数据备份的方法，即

使用 UNLOAD Utility 对某一时点的数据表进行数据摘取。UNLOAD 可以从若干个数据源 UNLOAD 数据到若干个 PS 文件中。数据源可以是 DB2 TABLESPACE 或者 DB2 IMAGECOPY 备份文件。

使用 UNLOAD，可以从整个 TABLESPACE 中 UNLOAD 记录或者从指定的 PARTITION 或 TABLE 上进行 UNLOAD。如果一个 TABLESPACE 是分 PARTITION 的，可以 UNLOAD 所有选定的 PARTITION 到一个数据集中，或者 UNLOAD 到多个数据集中。

关于 UNLOAD Utility 的相关参数，请参考 DB2 Utility 的相关章节。

在使用 UNLOAD Utility 对数据表中的数据进行摘取后，可以将这些数据通过 LOAD Utility 导入到源表或者其他数据表（表结构需要和源表完全一致）。

6.1.4 DB2 恢复工具及使用方法

DB2 数据表的恢复主要有两类，第一类是恢复某一张数据表某一时刻的备份，用于数据查询或其他用途。第二类是由于误操作或其他原因导致 DB2 表的数据遭到破坏需要恢复。而对应以上两种类型的恢复需求，日常维护中一般使用 DSN1COPY 和 RECOVER 两个 Utility 来进行恢复。DSN1COPY 主要用于第一类需求的数据恢复，RECOVER 主要用于第二类需求的数据恢复。

6.1.4.1 DSN1COPY Utility

DSN1COPY 是 DB2 的一种 OFFLINE Utility，即在数据库或表空间等没有启动的情况下也可以执行。DSN1COPY 主要用于各种类型文件的复制，具体有如下几类：

- 将 DB2 VSAM 文件复制到一个 PS 文件；
- 将一个 PS 文件复制到 DB2 VSAM 文件；
- 将 DB2 IMAGE COPY 文件复制到 DB2 VSAM 文件；
- 在 DB2 VSAM 文件之间互相复制；
- 将一个 PS 文件复制到另一个 PS 文件。

在进行数据恢复的时候，我们使用的是上面第三点功能：将 DB2 IMAGE COPY 文件复制到 DB2 VSAM 文件。需要注意的是数据库表空间在磁盘上存在实体就是 VSAM 文件，因此在使用 DSN1COPY 进行恢复时，就是将备份文件，即 IMAGECOPY 文件，以 PAGE 为单位复制到数据库表空间对应的实体 VSAM 文件。

使用 DSN1COPY 进行数据恢复的时候，需要根据备份数据源选择恢复方式。如果备

份源只有表空间，那么只对表空间对应的 VSAM 文件进行复制，具体步骤如下：

STEP 01 停止需要恢复的表空间。

STEP 02 使用 DSN1COPY 进行恢复，将备份文件恢复到目标表空间对应的 VSAM 文件。

STEP 03 启动需要恢复的表空间。

STEP 04 进行建索引操作。

如果备份源包括表空间和索引空间，那么需要对表空间和索引空间对应的 VSAM 文件分别进行复制，具体步骤如下：

STEP 01 停止需要恢复的表空间和索引空间。

STEP 02 使用 DSN1COPY 进行恢复，将备份文件恢复到目标表空间及索引空间对应的 VSAM 文件。

STEP 03 启动需要恢复的表空间和索引空间。

在使用 DSN1COPY 进行数据恢复之前，我们需要明确以下几个方面：

1．备份源文件

备份源文件就是 IMAGECOPY 文件，恢复之前需要确认要恢复的备份是哪天的哪个时间点的备份，是全量备份还是增量备份。明确恢复需求是保证数据恢复正确性的前提条件。

2．目标表

需要确定恢复目标表名称是否正确，一旦执行，那么表中当前的数据将被覆盖。

3．表结构

数据恢复之前需要确认备份源的表结构与需要恢复到的目标表的表结构是否一致。如果存在表结构的差异，恢复过程将会异常中断。

`DSN1COPY Utility` 的相关参数，请参考 `DB2 Utility` 的相关章节。

6.1.4.2　DB2 RECOVER Utility

DB2 RECOVER Utility 可以将数据表恢复到当前状态或以前的某个时间点，主要应用于第二类恢复需求，即由于出现误操作或其他原因导致 DB2 表的数据遭到破坏，需要恢复到某一正确的时间点。RECOVER 的对象最大的是 TABLESPACE 或 INDEXSPACE，最小的是 PAGE。对于 TABLESPACE，我们可以恢复整个 TABLESPACE 或某个 PARTITION，也可以是一段 PAGES 或某个单独的 PAGE。

RECOVER 的基本工作原理是 DB2 会根据恢复需求找到距离恢复时间点最近一次可用的全量备份，使用该备份恢复到备份点，再通过 Apply log 机制，将从备份点到恢复点间日志中记录追加到该表以完成该表的恢复。

如果 RECOVER 指定的 FULL IMAGECOPY 文件丢失，DB2 会向前寻找可用的 FULL IMAGECOPY 文件来恢复当前的 TABLESPACE 或 INDEXSPACE。

当使用了 TOLOGPOINT、TORBA、TOCOPY 等用于恢复到某个时间点的控制语句时，RECOVER 会将 INDEX 置为 REBUILD-PENDING 的状态。用户需要运行 REBUILD INDEX 作业清除 IDNEX 的 REBUILD-PENDING 状态。如果对有关联关系的表所在的 TABLESPACE 进行 RECOVER，务必对整个 TABLESPACE 进行 RECOVER，并对所有的 INDEX 进行 REBUILD，否则 DB2 会在 TABLESPACE 上置 CHECK-PENDING 状态。

在执行 RECOVER Utility 进行恢复工作前，建议先执行 REPORT Utility，通过执行 REPORT Utility，可以打印出需要恢复的表在 SYSCOPY、SYSLGRNX 以及 ARCHIVE LOG 中的相关信息，帮助用户迅速确定恢复的关键信息，提高恢复效率。

具体控制语句如下：

```
REPORT RECOVERY TABLESPACE  NASEDB.NSSJNL1
```

RECOVER Utility 的相关参数请参考 Utility 相关章节。

下面通过实例介绍几种常用的 RECOVER 案例。

1．恢复到某一点

提出此类需求，通常是对某张表执行某类误操作，导致表的内容发生改变，需要恢复到执行此项操作前某一时刻的表状态。

（1）TOCOPY——恢复到某一个 IMGCOPY 点

控制语句为：

```
RECOVER TABLESPACE NASEDB.NSSJNL1
TOCOPY BJUNLD.BB01.NASEDB.NSSJNL1.D0602010
```

（2）TOLOGPOINT——恢复到某一日志点

步骤如下：

STEP 01 用 DSN1LOGP 打印包含执行误操作所在 DB2 MEMBER 的 LOG 内容，找到在其中执行误操作那一刻的 STARTLRSN 值。

STEP 02 执行 RECOVER Utility。

控制语句为：

```
RECOVER  TABLESPACE NASEDB.NSSJNL1
TOLOGPOINT X'BE64178F486A'
```

X'BE64178F486A'就是通过 DSN1LOGP 查找到的 STARTLRSN 值。

（3）TORBA——在 NON-DATASHAING 环境中也可以指定 TORBA 参数，恢复到日志某一点。操作方法同 TOLOGPOINT。

注意：在恢复作业正确执行完毕后，该表的所有 INDEX 均处于 REBUILD PENDING 状态，需要对所有 INDEX 执行 REBUILD INDEX。相关联表也会处于 CHECK PENDING，需要做 CHECK。

2．恢复到当前点

提出此类需求，通常是对某张表执行某类操作（如 REORG 等），异常中断，导致表的状态和数据发生改变，需要通过做 RECOVER 将表恢复到当前点。

控制语句为：

```
RECOVER  TABLESPACE NASEDB.NSSJNL1
```

注意：在恢复作业正确执行完毕后，该表的所有 INDEX 均处于 REBUILD PENDING 状态，需要对所有 INDEX 执行 REBUILD INDEX。相关联表也会处于 CHECK PENDING，需要做 CHECK。

6.1.5　DB2 数据库备份策略参考

下面提供一个 DB2 例行数据备份策略的例子供读者参考。

6.1.5.1　DB2 系统表备份策略

DB2 系统表是指 DSNDB01 和 DSNDB06 数据库中的表，主要包括表空间如表 6.1 所示。

表 6.1　DB2 系统表

DSNDB01	DSNDB06
SCT02	SYSCOPY
SPT01	SYSDBASE
SYSLGRNX Log range	SYSDBAUT
SYSUTILX System utilities	SYSGPAUT

续表

DSNDB01	DSNDB06
DBD01	SYSGROUP
	SYSPKAGE
	SYSPLAN
	SYSSTATS
	SYSSTR
	SYSUSER
	SYSVIEWS

以上系统表对于 DB2 系统来说至关重要，DB2 系统表的正常可用是数据库系统正常运转的前提条件。

对 DB2 系统表的备份建议使用 DB2 COPY Utility 进行，建议每日在特定时间点进行全量备份并在备份执行前对 DSNDB01 和 DSNDB06 执行 QUIESCE Utility 取得一致点，以保证备份数据文件的一致性，并保留 31 天。

6.1.5.2 DB2 应用表备份策略

DB2 应用表是指用于业务处理的数据库表，对 DB2 应用表的备份建议使用 DB2 COPY Utility 进行，备份频率需要根据应用需求进行制定。在资源充裕的情况下，建议每日进行全量备份（FULL COPY），并采用联机方式进行备份，以减少对数据库连续性的影响。如果磁带库等资源比较紧张，可以考虑每周挑选一天进行全量备份（FULL COPY），而一周内的其余天进行增量备份（INCREMENTAL COPY）。备份介质保留时间为 31 天，与 DB2 ARCHIVE LOG 的保留日期保持一致，以保证一旦数据出现异常可以使用 DB2 恢复工具进行恢复。

6.2 DB2 日常健康检查

DB2 日常健康检查主要是对 DB2 系统各层面进行检查，也是保证 DB2 系统正常稳定运行的前提。本节将介绍几种基本健康检查项目，并通过实际案例进行分析讲解，帮助大家掌握 DB2 健康检查的基本方法。

6.2.1　DB2 系统级检查

6.2.1.1　DB2 地址空间检查

DB2 系统的地址空间在前面已经进行了介绍，主要包括 MSTR、IRLM、DBM1、SPAS 及 DIST 等，日常的健康检查首先要查看主要的几个地址空间是否正常启动，具体查看方法如下。

在 SDSF 的 DA 队列里面检查各地址空间是否处于启动状态：

```
PREFIX=PB11*  DEST=(ALL)  OWNER=** SORT=CPU%/D  SYSNAME=**
NP   JOBNAME  StepName   ProcStep    JobID     Owner  C  SysName
     PB11IRLM  PB11IRLM            S0173803  STCUSER    PT11
     PB11MSTR  PB11MSTR  IEFPROC  S0173779  STCUSER    PT11
     PB11DBM1  PB11DBM1  IEFPROC  S0173829  STCUSER    PT11
     PB11DIST   PB11DIST   IEFPROC  S0173837   STCUSER   PT11
```

如果有地址空间处于 ABEND 或 STOP 状态，则需要及时启动。

完成地址空间启动情况检查后，还需要进入 MSTR 地址空间，查看 JESMSGLG 中的内容，检查是否存在异常信息。常见的包括死锁（DEADLOCK）和超时（TIMEOUT）、资源不可用（RESOURCE UNAVAILABLE）等。

6.2.1.2　数据系统日志使用情况

DB2 的日志文件，简称 LOG，是 DB2 正常运转的关键。DB2 在发生任何更新、删除、插入等操作的时候，都在日志中记录。前面的章节已经对日志的原理进行了介绍，DB2 首先在 ACIVE LOG 中记录，写满一个后继续写下一个 ACTIVE LOG，ACTIVE LOG 可以定义多个（下面的例子中定义了 32 个）。

使用 DISPLAY LOG 命令来查看 DB2 日志使用情况：

```
--PB11 DIS LOG
--PB11 DIS LOG
 DSNJ370I  -PB11 DSNJC00A LOG DISPLAY
 CURRENT COPY1 LOG = DSNPB01.PB11.LOGCOPY1.DS10 IS 97% FULL
 CURRENT COPY2 LOG = DSNPB01.PB11.LOGCOPY2.DS10 IS 97% FULL
          H/W RBA = 0A8520016F5D
          H/O RBA = 0A84F5726FFF
          FULL LOGS TO OFFLOAD = 0 OF 32
          OFFLOAD TASK IS (AVAILABLE)
```

```
DSNJ371I  -PB11 DB2 RESTARTED 08:47:00 JUL 26, 2009
        RESTART RBA 0A6F226BF000
        CHECKPOINT FREQUENCY 5 MINUTES
        LAST SYSTEM CHECKPOINT TAKEN 14:59:44 AUG 12, 2009
DSN9022I  -PB11 DSNJC001 '-DIS LOG' NORMAL COMPLETION
```

本例中的命令查看了 PB11 子系统的日志使用情况，其中当前 ACTIVE LOG 使用率为 97%，没有等待 OFFLOAD 的 ACTIVE LOG（FULL LOGS TO OFFLOAD = 0 OF 32）。本例中，如果 FULLLOGS 的数量接近 32，说明最近一段时间 DB2 活动频繁，系统日志更新迅速，需要重点关注。

注意：每一个 DB2 MEMBER 都需要进行日志的检查工作。

6.2.1.3 DB2 表状态检查

DB2 表的正常状态为 RW，即 READ & WRITE，通过 DIS DB(*) SP(*) RES LIMIT(*) 命令，我们可以获取所有状态异常的表空间信息。如果表空间的状态不是 RW，需要关注并分析处理。

使用命令查看异常状态的 DB2 表空间，结果示例如下：

```
          --PB11 DIS DB(*) SP(*) RES LIMIT(*)
S0173779   DSNT360I  -PB11 **********************************
S0173779   DSNT361I  -PB11 *  DISPLAY DATABASE SUMMARY
                          *    RESTRICTED
S0173779   DSNT360I  -PB11 **********************************
S0173779   DSNT362I  -PB11    DATABASE = DSNDB04  STATUS = RW
                         DBD LENGTH = 262460
S0173779   DSNT397I  -PB11
          NAME     TYPE PART  STATUS          PHYERRLO PHYERRHI CA
          -------- ---- ----- ----------------- -------- -------- --
          SYSCOPY  TS        RW,COPY
   ******* DISPLAY OF DATABASE DSNDB04  ENDED     **********
S0173779   DSNT360I  -PB11 **********************************
S0173779   DSNT362I  -PB11    DATABASE = BCASDB  STATUS = RW
                         DBD LENGTH = 819704
S0173779   DSNT397I  -PB11
          NAME     TYPE PART  STATUS          PHYERRLO PHYERRHI CA
          -------- ---- ----- ----------------- -------- -------- --

          BSCRDTH  TS    0029 RW,COPY
          BSRJNL1  TS        RW,COPY
```

各种状态解释如下：

```
ACHKP    对象处于建议检查状态
CHKP     对象处于 CHECK-pending 状态
COPY     对象处于 COPY-pending 状态
GRECP    对象处于 group buffer pool RECOVER-pending 状态
LPL      显示处于 logical page list 的条目
RBDP     索引对象处于 REBUILD-或 RECOVER-pending 状态，包含了 REBP，LPL,WEPR 的状态。
RECP     对象处于 RECOVER-pending 状态，包含了 RECP, RECP*, LPL,和 WEPR (write error page
          Range 写页面错误范围).
REORP    对象处于 REORG-pending 状态.
RO        对象处于只读状态.
STOP     对象被停止，包括 STOP, STOPE, STOPP, and LSTOP 等状态.
UT       对象处于只允许 Utility 访问状态.
UTRO     对象处于允许 Utility 访问和只读状态.
UTRW     对象处于允许 Utility 访问和读写状态
UTUT     一个 Utility 正在运行在被检查对象，并且该对象只允许 Utility 访问
WEPR     显示写页面错误的信息.
```

异常状态的解决方法见第五节。

6.2.1.4　DB2 线程检查

DB2 线程的检查，主要是查看 DB2 子系统中所有线程的运行情况，根据实际情况处理异常线程。

使用“DIS THREAD(*)”命令进行 DB2 线程检查，结果示例如下所示：

```
DSNV401I  -PB11 DISPLAY THREAD REPORT FOLLOWS -
DSNV402I  -PB11 ACTIVE THREADS -
NAME     ST A   REQ ID        AUTHID   PLAN     ASID TOKEN
SERVER   RA *    13 db2jcc_appli BJB0006  DISTSERV 00E5 17607
 V437-WORKSTATION=dccbzhaohzz, USERID=BJB0006,
     APPLICATION NAME=db2jcc_application
 V445-L30ABF59.GD8B.C4B2A44AC62B=17607 ACCESSING DATA FOR 83.10.191.89
SERVER   RA *    73 db2jcc_appli BJA0066  DISTSERV 00E5 103863
 V437-WORKSTATION=dccbzhangqp, USERID=BJA0066,
     APPLICATION NAME=db2jcc_application
 V445-L30ABF60.GF2A.C4B26A17D18A=103863 ACCESSING DATA FOR
     83.10.191.96
TSO      T  *     3 BJS0024     BJS0024          011E 38732
RRSAF    T    15243            CANUSR   KO2PLAN  00DA 12914
```

```
RRSAF    T    5357            CANUSR   KO2PLAN  00DA 12915
RRSAF    T    2846            CANUSR           00DA 12930
RRSAF    T   2943 OMEGAMON    CANUSR   KO2PLAN  00DA 13073
CI11PE12 N       3             STCUSER          0167    0
CI11PA11 N       3             STCUSER          00E8    0
CI11PA11 ND   15156 ENTR28470001 NGDBA             00E8    0
CI11PA11 ND     544 ENTRMA310002 NGDBA            00E8    0
CI11PA11 ND   28526 ENTRMJ100003 NGDBA             00E8    0
```

DB2 的线程有 ACTIVE、INACTIVE、INDOUBT、POSTPONED 四种状态。可以通过指定 TYPE 来查看某种特定状态的 THREAD，也可以通过指定 SCOPE 来查看某个 DB2 GROUP 或 MEMBER 中 THREAD 的状态。

例如：

```
DIS THREAD（*） TYPE（INDOUBT） SCOPE（GROUP）
```

四种状态的含义如下。

- ACTIVE 状态：正在运行的 THREAD。
- INACTIVE 状态：通过 VTAM 与其他系统连接的 THREAD 正在等待一个新的工作单元请求。
- INDOUBT 状态：DB2 线程运行遵循 TWO-PHASE COMMIT 的原则，某线程在完成了第一个 PHASE 的 COMMIT 之后，与 COMMIT 的 COORDINATOR 失去了连接，它不知道是否应继续执行 COMMIT 或者是执行 ROLL BACK，就处于 INDOUBT 状态。处于 INDOUBT 状态的线程会持有资源的锁不释放，这样会影响其他线程对该资源的正常访问。因此需要及时处理处于 INDOUBT 状态的线程。对于 INDOUBT THREAD 的处理方法是使用“-RECOVER INDOUBT”命令对线程进行恢复。

```
-RECOVER INDOUBT(connid) ID（correlation-id）ACTION(xx)
```

此命令用于恢复 INDOUBT THREAD，其中 ACTION 的选项有 COMMIT 和 ABORT，一般我们都选择 ABORT 命令来恢复其 THREAD。

- POSTPONED 状态：某个工作单元进行 BACK-OUT 处理的动作被延迟了。对于 POSTPONED THREAD 的处理方法是使用“-RECOVER POSTPONED”命令将线程取消。

```
-RECOVER POSTPONED CANCEL
```

6.2.1.5　DB2 WLM状态

DB2 在进行远程存储过程调用的时候，通常使用 WLM 的方式进行，而 WLM 状态的正常是实现存储过程远程调用的前提。

使用如下命令检查 DB2WLM 状态：

```
D WLM,APPLENV=PB01WLM
```

结果如下：

```
-D WLM,APPLENV=PB01WLM
 IWM029I  15.56.03  WLM DISPLAY 368
   APPLICATION ENVIRONMENT NAME    STATE    STATE DATA
   PB01WLM                         AVAILABLE
   ATTRIBUTES: PROC=PB01WLM  SUBSYSTEM TYPE: DB2
```

其中 PB01WLM 是在 z/OS 的 WLM 组件中预先定义的应用环境（Application Environment），如果状态不是 AVAILABLE，需要使用如下命令进行启动：

```
V WLM,APPLENV=PB01WLM,Q
```

6.2.1.6　存储过程检查

使用命令：“-DIS PROC(*.*) SCOPE(GROUP)”来检查是否有被 STOP 的 PROCEDURE。

```
--PB11 DIS PROC(*.*) SCOPE(GROUP)
 DSNX940I  -PB11 DSNX9DIS DISPLAY PROCEDURE REPORT FOLLOWS -

 ------- SCHEMA=NGDBA
 PROCEDURE  STATUS ACTIVE QUED MAXQ TIMEOUT FAIL WLM_ENV
 CSTP00A
             STARTED    0    0    0      0    0 PB01WLM
 CSTP01A
             STARTED    0    0    1      0    0 PB01WLM

 ------- SCHEMA=SYSIBM
 PROCEDURE  STATUS ACTIVE QUED MAXQ TIMEOUT FAIL WLM_ENV
 SQLPROCEDURECOLS
             STARTED    0    0    1      0    0 PB01WLM4
```

结果需显示为无存储过程或者全是 Active 的。如果存在 STOP 状态的存储过程，需要使用如下命令启动：

```
--PB11 START PROC（NGDBA.XXXXX） SCOPE（GROUP）
```

6.2.1.7 DDF 状态检查

DDF（Distributed Data Facility，分布式数据访问组件）是 DB2 UDB for z/OS 的一部分。DB2 应用能够利用 DDF 访问位于其他 DB2 系统或者远程关系数据库系统中的数据。

使用-DISPLAY DDF 命令检查 DB2DDL 状态：

```
--PB11 DIS DDF
 DSNL080I  -PB11 DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
 DSNL081I STATUS=STARTD
 DSNL082I LOCATION          LUNAME          GENERICLU
 DSNL083I DSNPB01           VTAM1.PB11LU     VTAM1.PB01GRP
 DSNL084I IPADDR         TCPPORT RESPORT
 DSNL085I 83.16.49.31     4461    4481
 DSNL086I SQL    DOMAIN=dsnpb01.plexpt1.icbc.com
 DSNL086I RESYNC DOMAIN=pb11lu.dsnpb01.plexpt1.icbc.com
 DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

检查 STATUS 是否为 STARTD，如果不是，需要使用启动命令进行启动：

```
--PB11 START DDF
```

如果启用了基于 TCPIP 的 DDF 功能，则 IP 地址和端口号等都必须有对应的数值。

6.2.1.8 DB2 Utility状态检查

DB2 的 Utility 状态出现异常，可能会导致 DB2 表状态出现异常进而影响表的正常使用，因此进行 Utility 的例行检查，可以排除由其状态异常所引起的表状态异常，提高 DB2 的可用性。

使用- PB11 DIS UTIL(*)命令检查 DB2 Utility 状态。

```
--PB11 DIS UTIL(*)
 DSNU105I  -PB11 DSNUGDIS - USERID = NGTN3
              MEMBER = PB11
              UTILID = PIMG5ED1
              PROCESSING Utility STATEMENT 1
              Utility = COPY
              PHASE = COPY   COUNT = 0
              NUMBER OF OBJECTS IN LIST = 229
```

```
                LAST OBJECT STARTED = 199
                STATUS = ACTIVE
DSNU111I  -PB11 DSNUGDIS - SUBPHASE = COPYR COUNT = 209132
DSN9022I  -PB11 DSNUGCCC '-DIS UTIL' NORMAL COMPLETION
```

注意检查每个 Utility 的状态（STATUS），如果是 STOPPED，需要进一步分析处理。

6.2.1.9　DB2 TRACE开启情况检查

DB2 的 TRACE 是用来收集和记录 DB2 子系统各类数据及相关事件，用于后续的性能分析。

使用命令：“-DIS TRACE”检查当前 DB2 子系统开启了哪些 TRACE。

```
--PB11 DIS TRACE(*)
 DSNW127I  -PB11 CURRENT TRACE ACTIVITY IS -
 TNO TYPE   CLASS       DEST QUAL
 01  STAT   01,03,04,05, SMF  NO
 01         06
 02  ACCTG  01,02,03,07, SMF  NO
 02         08
 03  MON    30          OP1  NO

 *********END OF DISPLAY TRACE SUMMARY DATA*********
 DSN9022I  -PB11 DSNWVCM1 '-DIS TRACE' NORMAL COMPLETION
```

选择开启哪类 TRACE 需要根据实际需求确定。一般情况下，STAT 只打开 1、3、4、5、6 几类，ACCT 只打开 1、2、3、7、8 几类，MON 只打开 1 类型。如果开启过多的 TRACE，会导致数据库子系统性能的下降。

6.2.2　数据可用性和应用程序检查

6.2.2.1　DB2 表空间和索引空间文件扩展情况检查

DB2 表空间及索引空间的实体是 VSAM 文件，VSAM 文件的有大小及扩展次数的限制，一旦超过了允许范围，VSAM 文件将不可用，对应的 DB2 表就无法进行使用，因此 DB2 表空间实体文件的例行检查不可缺少。

VSAM 文件的扩展次数为 256 次，每日的健康检查建议对扩展次数超过 100 次的文件特别关注。对于扩展次数大于 100 次的表空间文件建议合适的时候安排空间定义调整，并重组以回收碎片空间，减少扩展次数。索引空间扩展次数大于 100 次的，考虑扩充空间后

进行表空间的联机重组，或者进行非联机的 REBUILD INDEX。

对于扩展次数的检查方法，可以根据实际情况进行考虑。对于 DB2 表较少且数据量很少的应用环境，可以在使用 UPDATE（ALL）参数进行 RUNSTATS 操作后，从 SYSIBM.SYSTABLEPART 里面查看 EXTENT 字段；对于 DB2 表较多，且数据量庞大的应用环境，建议使用操作系统 IDCAMS 的 DCOLLECT 功能收集 DB2 相关 VSAM 文件信息，再进行筛选。

6.2.2.2 DB2 表空间和索引空间文件大小检查

DB2 表和索引空间对应的 VSAM 文件有空间大小的限制，对于不同类型的表空间，限制有所不同，要依赖于当前系统环境配置和 DB2 表空间定义。例如使用 Large 定义的分区表空间在不使用 DSSSIZE 参数情况下最大为 4GB，约 87000 TRACKS。每日的健康检查建议将文件大小接近 60000 TRACKS 的文件特别关注，如果继续呈增长趋势，则需要考虑压缩或者重新调整 Partition 键值。

6.2.2.3 DB2 相关Storage Group 使用率检查

DB2 数据表实体都是存在于磁盘上的文件，在使用 SMS 管理的环境中，可根据不同的 DB2 文件类型对应的访问方式不同，划分不同的 Storage Group，从系统配置层面进行不同的设置达到分别管理的目的。例如应用 DB2 可以根据需要放在一个或多个 Storage Group 中，DB2 BSDS、Active Log 及 Archive Log 习惯使用单独的 Storage Group。

每日健康检查的时候，需要关注各个 DB2 相关的 Storage Group 的使用率，对于使用率超过 80%的 Storage Group，需要关注并及时增加磁盘卷。

6.2.2.4 应用程序访问路径检查

通常在未对程序进行 REBIND 或者 BIND 时，程序的访问路径是不会变化的。但是为了防止在日常工作中，对表进行了重建和 REBIND 等操作后访问路径发生变化，建议定期进行访问路径的检查确认。

访问路径的检查主要是访问 PLAN_TABLE 表，比较前后两次执行 REBIND 或 BIND 后程序的访问路径，找出变化的信息。分析该变化是选择正确还是错误的访问路径。

6.2.2.5 INVALID PACKAGE检查

PACKAGE，也就是程序，在正常情况下的状态应为 VALID，即可用状态。如果程序

的状态为 INVALID，则代表此程序失效，如果在联机或者批量处理过程中调用到失效的程序将会报错，因此需要保证所有程序的状态为 VALID。

如果发生了如下变化，则程序的状态将自动变为不可用（INVALID）：

- 程序依赖的 TABLESPACE 被 DROP；
- ALTER TABLE ADD COLUMN，增加的字段类是 DATA，TIME，TIMESTAMP 时；
- 程序依赖的 INDEX 被 DROP；
- 程序依赖的 PROCEDURE 被 DROP。

使用如下 SQL 检查 INVALID PACKAGE 的数目。

```
SELECT * FROM SYSIBM.SYSPACKAGE WHERE VALID = 'N';
```

当程序状态为 INVALID 的时候，需要使用 REBIND 命令来使失效的程序变为可用（VALID）。

REBIND 是根据最新的 DB2 表空间及索引的统计信。生成最新的执行模块（带有最新的访问路径）。

REBIND 作业范例如下：

```
//JOBLIB  DD  DSN=DSN810.SDSNLOAD,DISP=SHR
//PH02CS04 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//REPORT   DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSTSIN DD *
   DSN SYSTEM(PB02)
   REBIND PACKAGE(PKGON.OAH5150) EXPLAIN(YES)
```

REBIND 作业正常结束后，程序的状态也由 INVALID 变为 VALID。需要说明的是，如果符合以上导致程序状态变为不可用的条件没有恢复，即使执行 REBIND，程序也被识为废弃的程序，可通过执行 Free Package 方法进行废弃程序的处理。

6.2.2.6　检查DB2 备份情况

数据是所有应用的基础，因此对应用数据库的备份是一项非常重要的工作，而完成备份后，确认备份数据的完整性也必不可少。

使用如下 SQL 检查全量备份是否完整：

```
SELECT * FROM SYSIBM.SYSTABLEPART A
WHERE NOT EXISTS(SELECT * FROM SYSIBM.SYSCOPY B
WHERE A.TSNAME = B.TSNAME AND A.DBNAME = B.DBNAME
AND B.ICDATE = 'YYMMDD' ' AND B.ICTYPE = 'F' )
AND A.DBNAME IN
('APPDB1','APPDB2'…….'APDBN');
```

根据需求修改 SQL 语句中的日期 YYMMDD 及应用数据库名称：APPDB1 等。

使用如下 SQL 语句检查增量备份是否完整：

```
SELECT * FROM SYSIBM.SYSTABLEPART A
 WHERE NOT EXISTS(SELECT * FROM SYSIBM.SYSCOPY B
WHERE A.TSNAME = B.TSNAME AND A.DBNAME = B.DBNAME
AND B.ICDATE = 'YYMMDD' AND B.ICTYPE = 'I' )
AND A.DBNAME IN
('APPDB1','APPDB2'…….'APDBN');
```

根据需求修改 SQL 语句中的日期 YYMMDD 及应用数据库名称：APPDB1 等。输出均必须为 0。如果不为 0，说明备份有遗漏的表，需要及时补充备份。



6.3 DB2 重组

DB2 重组工作是 DB2 日常维护工作的重点，也是提高 DB2 整体访问性能的最佳手段，本节将结合实际介绍 DB2 重组的相关问题。

6.3.1 进行DB2 重组的目的

1. 调整或回收DB2 空间

对于数据量持续增加的数据表，其表空间及索引空间对应的 VSAM 文件的大小不断增长，导致文件的扩展次数增加，如果持续增加到 256 次，数据表将不可用。而从程序访问角度来说，VSAM 文件扩展次数的增加也在一定程度上影响程序的访问效率。

而对于目前占用磁盘较大，但是实际数据量却较少的数据表，通过重组的方式也可以达到回收磁盘空间的目的。

2．使DB2 数据排列整齐有序，提高程序访问效率

通过重组，可以提高数据表中的数据排列的整齐程度，为程序选择最佳的访问路径提供依据。

6.3.2　DB2 REORG的对象

根据对象的不同，重组可以分为对表空间的重组和对索引空间的重组。表空间的重组可以分为整表重组和按分区（PARTITION）重组。

索引空间的重组也可按索引进行重组和按分区（PARTITION）进行重组，如表 6.2 所示。

表 6.2　重组的分类

重组表空间	By Tablespace
	By Partition
重组索引空间	By Index
	By Partition

虽然重组可以增加数据的整齐度，提高程序的访问效率，但是并不是所有的表都需要进行重组的表，下面几种类型的表就不需要定期进行重组。

1．每天进行清空操作的表

由于应用的需要，可能某些数据表每日都使用 LOAD DUMMY 的方式清空当前数据，然后再重新写入数据。由于 LOAD DUMMY 操作已经进行了空间的释放动作，因此重组这类表空间并没有实际意义。

2．数据量稳定的表

在实际应用中，部分应用表数据量很小，甚至只有 1 条记录，而且基本不会新增、更改和删除记录，所以这类表没有必要做重组。

6.3.3　DB2 重组的条件

如何判断哪些表需要重组，是我们进行重组的前提。对于不同的 DB2 应用，数据表的重组频率是不同的，因为不同的数据需要不同的重组规划，这些规划取决于下面的一些因素：

- 修改活动的频率（插入、更新及删除）；

- 应用的交易量；
- 创建表空间或索引空间时候被分配的空闲空间量。

从 DB2 文件存储空间角度来讲，建议将 EXTEND 次数偏高的表空间或索引空间进行重组；从应用访问的角度来讲，建议将程序访问性能差的表空间和索引空间进行重组。

根据经验将上面的重组条件具体化，我们得出表空间及索引空间进行重组的细化条件。

6.3.3.1 表空间进行重组的条件

- SYSIBM.SYSINDEXPART 表中的 FAROFFPOSF / CARDF > 10%；
- 如果索引是 Clustering 索引，这个索引的 CLUSTERRATIOF 值< 90%；
- SYSIBM.SYSINDEXPART 表中(NEARINDREF + FARINDREF) / CARDF> 10%；
- 简单表空间（SIMPLE TABLESPACE）的 PERCDROP> 10% ；
- 在使用 ALTER TABLE 命令对数据表进行修改后，表处于 AREO* 状态；
- 表空间对应的 VSAM 文件的扩展次数超过 100 次。

6.3.3.2 索引进行重组的条件

- SYSIBM.SYSINDEXPART 表中的 LEAFFAR / NLEAF >10%；
- SYSIBM.SYSINDEXPART 表中的 PSEUDO_DEL_ENTRIES / CARDF > 10%；
- 大量数据的删除操作后；
- 在使用 ALTER INDEX 命令对索引进行修改后，索引空间处于 AREO 或者 ARBDP 状态的时候；
- 索引对应的 VSAM 文件的扩展次数超过 100 次。

6.3.4 DB2 重组的方法

根据重组时数据表是否可以提供正确访问，我们将重组分为在线重组（ONLINE REORG）和离线重组（OFFLINE REORG），其中在线重组一般称为联机重组，意思是重组可以与应用访问并行。而离线重组还有一种特殊方式，即 DIAGNOSE ABEND 方式的重组。

（1）OFFLINE REORG 方法：使用 SHARE LEVEL NONE 的参数，在 REORG 时表不可访问，一般用于频繁更新的表空间。

（2）ONLINE REORG：SHARE LEVEL CHANGE/REFERENCE，REORG 时允许读写或只读访问，日常 REORG 尽量都使用 ONLINE REORG，以减少对应用的影响。

在使用 ONLINE REORG 的时候，还需要提前建立若干 MAPPING TABLE 配合重组作业，MAPPING TABLE 的 DDL 如下：

```
CREATE TABLESPACE MAPTS1
       IN MAPDB
       USING STOGROUP SGPDTB01
            PRIQTY 5000
            SECQTY 1000
            ERASE NO
       FREEPAGE 3
       PCTFREE 5
       SEGSIZE 32
       BUFFERPOOL BP0
       LOCKSIZE PAGE
       LOCKMAX 0
       CLOSE YES
       LOCKPART NO
       MAXROWS 255
;
CREATE TABLE NGDBA.MAP_ROG1
      (
       TYPE   CHAR(1)  FOR MIXED DATA NOT NULL ,
SOURCE_RID  CHAR(5)  FOR MIXED DATA NOT NULL ,
TARGET_XRID  CHAR(9)  FOR MIXED DATA NOT NULL,
      LRSN      CHAR(6)   FOR MIXED DATA NOT NULL
      )
      IN MAPDB.MAPTS1                                    ;
CREATE UNIQUE INDEX NGDBA.XMAP_ROG1
      ON NGDBA.MAP_ROG1
      (SOURCE_RID    ASC
      ,TYPE         ASC
      ,TARGET_XRID   ASC
      ,LRSN         ASC
      )
      USING STOGROUP SGPDTB01
           PRIQTY 5000
           SECQTY 5000
           ERASE NO
      FREEPAGE 0
      PCTFREE 10
      BUFFERPOOL BP0
      CLOSE NO
      PIECESIZE 2097152 K                                 ;
```

另外，在进行联机重组的时候，一个重组作业需要使用一个独立的 MAPPING TABLE，因此如果需要并行运行多个联机重组作业的时候，需要提前定义足够的 MAPPING TABLE。

DIAGNOSE REORG：通过 DIAGNOSE 语句控制，先完成数据的 Reload PHASE，再通过单独的作业进行全部索引 REBUILD 工作。适用于有多个索引的大表。

如果数据表符合以下几点，就需要使用 DIAGNOSE 方式进行重组：

- 数据表的数据量很大；
- DB2 表空间是分 PART 的；
- 数据表有多个 NPI。

DIAGNOSE 方式重组的步骤如下（U300 是 RELOAD PHASE 阶段完成的标志信息）：

STEP 01 数据 UNLOAD。

STEP 02 数据 RELOAD，其中 U300 的信息是完成 RELOAD 的标志。

STEP 03 DB2 监控到 U300 的信息后，将作业终止。

STEP 04 检查所有作业是否都已正常完成 RELOAD。

STEP 05 使用“TERM UTIL(XXX)”的命令就 STOPPED 的 Utility 终止。

STEP 06 使用 REBUILD 作业对数据表相关索引进行 BUILD 操作。

```
DIAGNOSE ABEND MESSAGE U300
  INSTANCE 1
  NODUMP
    REORG TABLESPACE PRASDB.PSFOSUB PART 84
      LOG NO COPYDDN(SYSCOPY)
      STATISTICS TABLE(ALL) INDEX(ALL) REPORT(YES)
      SHRLEVEL NONE
DIAGNOSE END
```

表 6.3 通过几个方面对比了 Online 和 Offline 重组的异同：

表 6.3　在线与离线重组的区别

重组方式	数据表的可用性	风　险	速　度	Share level
ONLINE	可用	低	慢	Change/Reference
OFFLINE	不	较高	快	None

6.3.5　DB2 重组的注意事项

- 在表空间与索引空间都需扩展重组时，要串行安排。

- 要提前预估重组过程的存储需求。
- 必须指定 LOG NO，否则 REORG Utility 在 RELOAD PHASE 写大量 LOG。
- 严禁重组过程中清除中间文件。
- 使用 START ACCESS(FORCE)时要小心，严禁在生产环境使用该命令。
- 重组运行时必须随时监控磁盘使用空间。

6.3.6　DB2 重组常见问题及解决方法

1．问题一：IMAGE COPY文件空间不足导致中断

【问题现象】

由于 IMAGE COPY 文件空间指定较小，作业异常中断（ABEND S04E），检查 Utility 状态时发现 Utility 处于 STOP 状态，表空间的状态异常。

【解决方法】

- 使用命令 TERM Utility（UtilityID）将 Utility 终止；
- 对已经生成的 IMAGE COPY 及 UNLOAD SYSREC 文件进行清理，防止在后续作业提交过程中出现 JCL ERROR；
- 修改作业中 IMAGE COPY 文件空间分配大小后，重新提交作业。

2．问题二：由于UNLOAD 文件空间不足导致中断

【问题现象】

由于 UNLOAD SYSREC 文件空间指定较小，作业异常中断（ABEND S04E），检查 Utility 状态时，发现 Utility 处于 STOP 状态，表空间的状态异常。

【解决方法】

- 使用命令 TERM Utility（UtilityID）将 Utility 终止；
- 对已经生成的 IMAGE COPY 及 UNLOAD SYSREC 文件进行清理，防止在后续作业提交过程中出现 JCL ERROR；
- 修改作业中 UNLOAD SYSREC 文件空间分配大小后，重新提交作业。

3．问题三：RELOAD阶段由于表空间不足导致中断

【问题现象】

在 RELOAD 阶段，由于 DB2 表空间对应的 VSAM 不可用，导致作业中断（ABEND S04E）。

而针对 VSAM 文件不可用，又可细分为以下两类：

（1）表空间对应的 VSAM 文件的扩展次数超过 256 次；

（2）表空间对应的 VSAM 文件大小超过 4GB。

【解决方法】

对于第一种情况要确认 UNLOAD 文件是否存在。通过 ALTER TABLESPACE 命令对表空间首分配及次分配的调整，根据表空间占用情况扩大首分配空间。

```
ALTER TABLESPACE APPDB1.APPSDS1 PART 1 PRIQTY  0 SECQTY 131;
```

对于 DB2 V7.1 版本以前的环境需要，在 Utility 参数处使用 RESTART（PHASE）进行重新提交。对于 DB2 V8.1 版本以上的环境无须修改作业，直接重新提交中断作业即可。

```
//ROGTBL  EXEC DSNUPROC,PARM='PB01,APP1，RESTART（PHASE）'
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//UTPRINT  DD  SYSOUT=*
//SORTOUT  DD  UNIT=(SYSDA,5),SPACE=(CYL,(2000,500),RLSE)
//SYSUT1   DD  UNIT=(SYSDA,5),SPACE=(CYL,(2000,500),RLSE)
//SORTWK01 DD  UNIT=SYSDA,SPACE=(CYL,(2000,500),RLSE)
//SORTWK02 DD  UNIT=SYSDA,SPACE=(CYL,(2000,500),RLSE)
//SORTWK03 DD  UNIT=SYSDA,SPACE=(CYL,(2000,500),RLSE)
//SORTWK04 DD  UNIT=SYSDA,SPACE=(CYL,(2000,500),RLSE)
```

对于第二种情况，则通过 ALTER TABELSPACE 命令将对应的表空间或者 PART 进行压缩，然后重新提交出错的作业。

```
ALTER TABLESPACE APPDB2.APPSDS2 PART  1  COMPRESS YES;
```

4．问题四：SORT或BUILD PHASE中断

【问题现象】

重组作业可能由于 SORT 空间不足、Mapping Table Index 空间不够大等原因，而造成重组作业在 SORT 或 BUILD 阶段中断。

【解决方法】

分析解决造成 SORT 或 BUILD 出错的具体原因；对于 SORT 空间不足的情况，重组作业中增加 SORTWRK 文件大小分配；对 MAP TABLE 空间不足的情况，扩大 MAP Table 空间。再重新提交重组作业，作业就可以直接从中断点继续执行。

5．问题五：LOG APPLY PHASE中断（ONLINE REORG）

【问题现象】

重组作业可能由于联机交易访问频繁等原因造成重组作业中断。

【解决方法】

由于我们在重组作业中已经通过指定 LONGLOG、RETRY、RETRY_DELAY 、DRAIN_WAIT、TIMEOUT 等参数进行相关控制，如果作业由于访问频繁等原因，此重组作业会自动中断，无须进行相关处理，重新提交作业即可。

如果由于其他原因造成中断，还需要按以下流程处理：

- 分析解决造成 LOG APPLY 出错的具体原因；
- 通过 DISLPAY Utility（*）命令，查看是否中断作业的 Utility 处于 STOPPED 状态；如果 Utility 处于 STOPPED 状态，发命令 TERM Utility（UtilityID）来处理；
- 确认重组表空间的状态，并检查 VSAM DATASET 名字是否发生改变，有没有 SHADOW DATASET 存在（表空间状态应正常，VSAM DATASET 名字应没有改变，SHADOW DATASET 也不存在）；
- 重新提交重组作业即可。

6．问题六：SWITCH和BUILD2 PHASE中断（ONLINE REORG）

【问题现象】

重组作业在 SWITCH 或 BUILD2 阶段发生中断。

【解决方法】

由于我们在重组作业中已经通过指定 TIMEOUT 或 DEADLINE 参数进行相关控制，如果达到以上条件，此重组作业会自动中断，无须进行相关处理，只能重新提交作业。

如果由于其他原因造成中断，并且重组作业 Utility 还在，作业是可以 RESTART 的，按以下流程处理：

- 分析解决造成 SWITCH 或 BUILD NPI 出错的具体原因；
- 通过 DISLPAY Utility（*）命令，查看是否中断作业的 Utility 处于 STOPPED 状态；
- 重新提交重组作业即可。

7．问题七：重定义后的表或索引空间不足

处理方法如下。

- 确认中断作业的 Utility ID，并进行 TERM 操作。
- 使用 ALTER 命令调整相关表空间及索引空间的首分配及次分配。
- 重新提交作业。



6.4 DB2 RUNSTATS

RUNSTATS 程序为 DB2 表、表空间，表分区、索引及数据表中的列收集统计信息。它可以把这些信息放置于 DB2 的 CATALOG 表中，或者生成一个统计信息的报表。这些统计信息主要用于以下两方面：为 DBA 提供组织信息、并且在 BIND 处理时作为 DB2 优化器的输入，以便决定程序的最佳访问路径；还可以协助 DBA 使用 SQL 来查询统计信息，了解 DB2 日常状态，掌握 DB2 运转信息。

6.4.1 定期执行RUNSTATS的目的

对于 DB2 管理员来说，定期对应用 DB2 进行 RUNSTATS 操作是非常必要的，其目的主要有以下几点：

- 便于 DBA 定期掌握 DB2 数据的组织情况，进行性能分析及调整计划（如重组等）的确定。

定期 RUNSTATS 可以获得最新的 DB2 表及索引的相关信息，比如索引的 CLUSTER RATIO 偏低的时候，需要进行重组；DB2 表或索引对应的 VSAM 文件扩展次数偏多，需要调整表空间及索引空间的首分配及次分配。

- DB2 日常运行过程中，数据量有所变化，通过定期 RUNSTAT 使 DB2 访问路径最佳。

定期 RUNSTATS 可以获得最新的 DB2 表及索引的相关信息，在程序进行 BIND 的时候，指引 DB2 选择最合适的索引，使程序访问路径达到最佳。

6.4.2 何时需要进行RUNSTATS操作

- 大规模数据恢复或数据抽取完成后；
- DB2 表的数据量产生突变后；
- 表结构变化或索引发生变化后。

6.4.3　RUNSTATS注意点

在对 DB2 执行 RUNSTATS 操作时，需要：

- 在执行 RUNSTATS 时采用 SHRLEVEL CHANGE 方式，不需要停止应用对 DB2 的访问；
- 在所有表完成 RUNSTATS 后，尽量安排进行相关程序的 REBIND；
- 对表和表所有的 INDEX 的 RUNSTATS 要同时进行，并注意加 KEYCARD 参数；
- 注意特殊表的处理（数据量变化频率较大），对于这些特殊表，由于可能存在数据量很少的情况，如果此时进行了 RUNSTATS 操作，可能导致程序选择错误的访问路径，因此执行 Runstats 要在数据量较多时进行；
- 在使用 UPDATE ACCESSPATH、UPDATE SPACE 及 UPDATE ALL 参数完成 RUNSTATS 操作后，要将所有访问相关表的程序进行 REBIND 操作，使程序根据最新的信息进行访问路径选择。



6.5　STOSPACE

STOSPACE 可以统计 DB2 表空间和索引空间对应的文件的分配大小及使用率，此 Utility 控制参数很简单：

```
STOSPACE    STOGROUP(stogroup-name)
```

其中 stogroup-name 是指建表及索引时指定的 Stogroup 名称。

STOSPACE 运行结束后，会对如下 DB2 CATALOG 表的部分字段进行更新，以下字段的单位均为 KB：

- SYSIBM.SYSINDEXES 表中的 SPACE 字段，如果索引空间定义不在对应的 STOGROUP 中或者没有运行 STOSPACE，则此字段为 0。
- SYSIBM.SYSTABLESPACE 表中的 SPACE 字段，如果表空间定义不在对应的 STOGROUP 中或者没有运行 STOSPACE，则此字段为 0。
- SYSIBM.SYSINDEXPART 表中的 SPACE 字段，如果索引空间的分区定义不在对应的 STOGROUP 中或者没有运行 STOSPACE，则此字段为 0。
- SYSIBM.SYSTABLEPART 表中的 SPACE 字段，如果表空间的分区定义不在对应的

STOGROUP 中或者没有运行 STOSPACE，则此字段为 0。

- SYSIBM.SYSSTOGROUP 表中的 SPACE 字段，代表所有使用相应 STOGROUP 的表空间及索引空间所分配的空间大小。
- SYSIBM.SYSSTOGROUP 中的 STATSTIME 字段，代表最近一次运行 STOSPACER 的时间。



6.6 DB2 ROTATE 操作

从 DB2 V8 版本开始，用户可以进行动态的分区增加，分区 ROTATE 及分区间的数据量自动均衡调整操作。

ROTATE 是指用户可以调整数据表的 LIMIT KEY，从而达到循环使用 DB2 现有分区的一种方法。例如一张应用表有 12 个分区，2009 年 1 月的数据使用第一个分区，2009 年 2 月的数据使用第二个分区，依此类推。此表的数据只需要保留 12 个月，那么在 2010 年 1 月的时候，可以使用 ROTATE，将 2009 年 1 月数据对应的 PART 的 LIMITKEY 进行更改，用来存放 2010 年 1 月的数据。

在进行 ROTATE 操作的时候，DB2 表状态为不可用，完成 ROTATE 操作后，需要对此表相关的所有程序进行 REBIND 操作。



6.7 DB2 性能监控及调整

作为数据库管理员，在数据库运行过程中，必须有规律地监控数据库的运转情况，发现影响数据库性能的问题，及时分析调整。在实际环境中，很多因素会影响数据库的性能，数据库管理员需要制定一套详细的，科学的、有规律地监控方法，并且真正运用到实际的数据库维护工作中，才能有效并及时地发现问题，解决问题。本节将从数据库的性能监控和性能调整两大方面介绍，帮助数据库管理员跟踪数据库的性能问题并及时进行优化。

6.7.1 DB2 性能监控

在进行数据库性能监控之前，必须对数据库的性能有一个明确的概念。一般来说，主

要有以下四个因素影响到数据库的性能。

1．工作负载

工作负载主要指那些连接到数据库的联机及批量的处理，它可能根据时间的变化发生极大的变化，所有的工作负载对数据库的性能都有很大的影响。

2．吞吐量

吞吐量定义了计算机处理数据的总的能力，它是由 I/O 速度、CPU 速度、操作系统的有效性组成的。

3．资源

系统进行处理时的硬件和软件工作被称为系统的资源，主要包括 CPU、内存、磁盘、高速缓存控制器及微码等。

4．竞争

当对一个特定资源的要求（工作负载）很高时，竞争便可能产生。竞争是指工作负载的两个或更多组件试图使用单个资源的情况。当竞争增加时，吞吐量会相应降低。

数据库性能监控主要可以从 TRACE 信息及报告、DB2 LOG 信息、DB2 资源状态、DB2 CATALOG 表和程序访问路径几个方面展开。

6.7.1.1　使用DB2 PM进行性能监控

DB2 PM（Performance Monitor）也称 DB2 PE（Performance Expert），作为 DB2 子系统性能分析工具，提供联机监控和批量报表，帮助数据库管理员评价 DB2 系统的性能和应用程序的性能，为数据库管理员提供系统优化和应用优化的依据和方法。使用 DB2 PM 进行数据库监控的前提是在需要监控的数据库上启动相应的 TRACE，命令如下：

```
-START  TRACE(XXXX)  CLASS(X,X,X,…)
```

其中 TRACE 分为 PERFM、ACCTG、STAT、AUDIT、MONITOR 五类，CLASS 根据不同的 TRACE 类型不同的数字表示 TRACE 需要跟踪的不同指标。具体请参考 DB2 COMMAND 相关章节。

- Performance Trace

此类 TRACE 记录了关于各类特定 DB2 事件信息，用于性能分析和调优，一般只在使用其他途径的监控和调整后，无法得到准确结论的情况下，才使用此类 TRACE，因为打开

Performance Trace 会消耗大量系统资源。

- Accounting Trace

Accounting Trace 主要用于分析及判断 DB2 应用程序的性能，通常是以一个线程为单位记录的。

- Statistics Trace

Statistics Trace 记录与整个 DB2 子系统相关的信息，是基于数据库系统级的。

- Auditing Trace

Auditing Trace 一般对 DB2 系统内各类审计类的追踪和记录。

- Monitor Trace

Monitor Trace 主要是对监控类信息进行收集以便提供数据给 DB2 监控产品。

以上这些 TRACE 产生的数据被写入到 GTF、SMF、RES、SRC、OP 和 OPX 这 6 个目标中。其中 SMF 为最常用的数据存储目标，大部分 DB2 PM 报告的原始信息也会写入 SMF。

1. 使用 DB2 PM 进行联机监控

在使用 DB2 PM 进行联机监控的时候，你可以将 PM 看作一个窗口，通过这个窗口，你可以看到 DB2 工作负载的性能特征。完成 PM 的配置并登录后，可以看到连接到你所配置的 DB2 的所有线程，以及各线程在应用及数据库中的时间消耗等各种指标信息，如图 6.1 所示。

2. 通过打印 DB2 PM 报告进行事后分析

除了使用 PM 的联机监控方式对数据库进行性能监控外，还可以使用 PM 打印 Performance Report，供 DB2 的性能进行事后分析。PM 报告常用的有 Accounting Report、Statistics Report、Locking Report 和 Audit Report 等，以上这些报告的数据源一般来自系统 SMF 数据。

（1）Accounting Report

Accounting Report 通常是以线程为单位记录此线程相关的信息，提供了诸如每个程序的启动和停止时间、每个程序的执行次数、使用的 SQL 语句的类型和每个类型执行的次数、BUFFER POOL 的使用情况、锁资源的使用情况、CPU 资源使用情况、同步 I/O 及异步 I/O 等待时间、RID POOL 使用情况等相关信息。

Accounting Report 是进行 DB2 应用程序性能分析最常用的手段之一。

Accounting Report 作业范例如下：

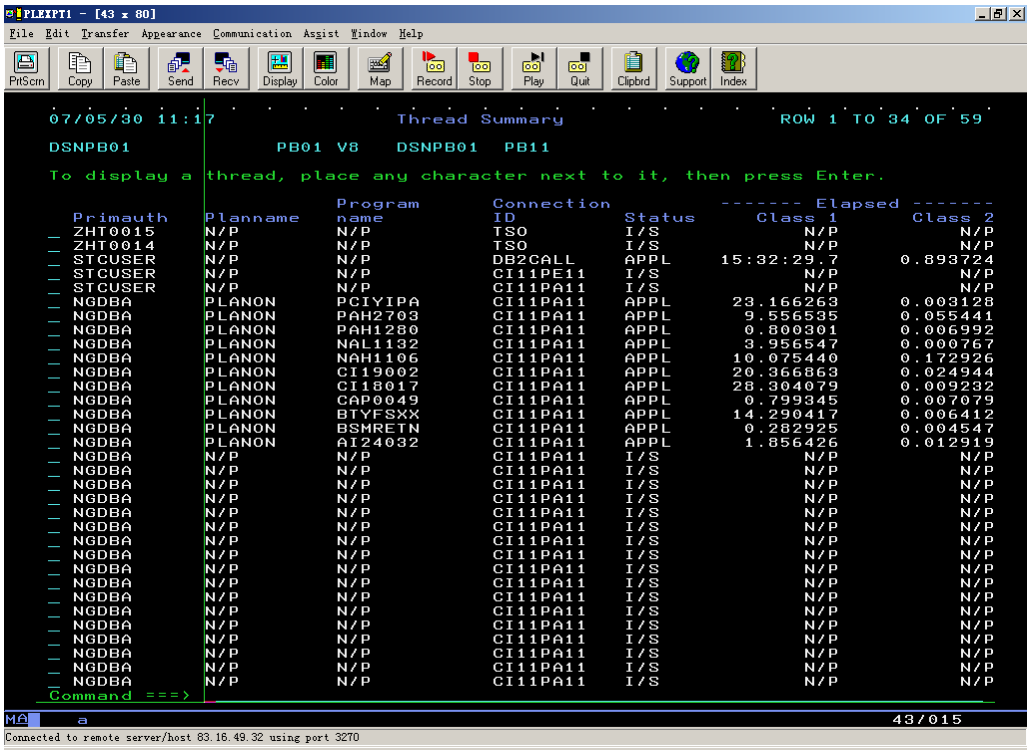


图 6.1　DB2 工作负载的性能特征

```
//DB2PMRPT EXEC PGM=DB2PM
//STEPLIB DD DSN=FPE.V2R1M0.SFPELOAD,DISP=SHR
//INPUTDD DD DISP=SHR,DSN=SMFMRGSH.PLEXPT1.DB2.D090904
//JOBSUMDD DD SYSOUT=*
//ACWORK   DD DSN=&&ACWK,
//        DCB=(RECFM=VBS,LRECL=32756,BLKSIZE=6233),
//        SPACE=(CYL,(1000,1000)),UNIT=(SYSDA,15)
//STWORK   DD DSN=&&STWK,
//        DCB=(RECFM=VBS,LRECL=32756,BLKSIZE=6233),
//        SPACE=(CYL,(1000,1000)),UNIT=(SYSDA,15)
//SORTWK01  DD   DISP=(NEW,DELETE),UNIT=SYSDA,
//               SPACE=(CYL,(500,900))
//SORTWK02  DD   DISP=(NEW,DELETE),UNIT=SYSDA,
//               SPACE=(CYL,(500,900))
//SORTWK03  DD   DISP=(NEW,DELETE),UNIT=SYSDA,
//               SPACE=(CYL,(500,900))
//SORTWK04  DD   DISP=(NEW,DELETE),UNIT=SYSDA,
//               SPACE=(CYL,(500,900))
```

```
//SORTWK05  DD   DISP=(NEW,DELETE),UNIT=SYSDA,
//                 SPACE=(CYL,(500,900))
//SORTWK06  DD   DISP=(NEW,DELETE),UNIT=SYSDA,
//                 SPACE=(CYL,(500,900))
//SORTWK07  DD   DISP=(NEW,DELETE),UNIT=SYSDA,
//                 SPACE=(CYL,(500,900))
//DPMOUTDD DD DUMMY
//DPMPARMS DD DSN=SHRLIB.BJS0021.DPMPARMS.LIB,DISP=SHR
//STRPTRD  DD SYSOUT=*
//SYSIN DD *
  GLOBAL(INCLUDE(SUBSYSTEMID(PB02)),TIMEZONE(08:00)
       FROM(06/09/09,13:55:00.00)
         TO(06/09/09,20:00:00.00)
       )
    ACCOUNTING
       REPORT
         LAYOUT(LONG)
          INCLUDE(CORRNAME(YXW12022),PROGRAM(YXKP0061))
  EXEC
```

（2）Statistics Report

Statistics Report 提供了关于 DB2 子系统级的性能信息，它显示了在一定时间段内，DB2 子系统所有活动的汇总，主要包括 SQL 执行信息、锁资源信息、RID POOL 信息、存储过程调用信息、数据库日志信息、EDM POOL 信息、Buffer Pool 信息等。

Statistics Report 作业范例如下：

```
//DB2PMRPT EXEC PGM=DB2PM
//STEPLIB DD DSN=FPE.V2R1M0.SFPELOAD,DISP=SHR
//INPUTDD DD DISP=SHR,DSN=SMFMRGSH.PLEXPT1.DB2.D080626
//JOBSUMDD DD SYSOUT=*
//ACWORK  DD  DSN=&&ACWK,
//          DCB=(RECFM=VBS,LRECL=32756,BLKSIZE=6233),
//         SPACE=(CYL,(1000,1000)),UNIT=(SYSDA,15)
//STWORK   DD DSN=&&STWK,
//           DCB=(RECFM=VBS,LRECL=32756,BLKSIZE=6233),
//          SPACE=(CYL,(1000,1000)),UNIT=(SYSDA,15)
//SORTWK01  DD   DISP=(NEW,DELETE),UNIT=SYSDA,
//            SPACE=(CYL,(500,300))
//SORTWK02  DD   DISP=(NEW,DELETE),UNIT=SYSDA,
//            SPACE=(CYL,(500,300))
//DPMOUTDD DD DUMMY
//STRPTRD  DD SYSOUT=*
```

```
//SYSIN DD *
GLOBAL(INCLUDE(SUBSYSTEMID(PB12,PB22,PB32,PB42)),TIMEZONE(08:00)
      FROM(06/26/08,16:20:00.00)
        TO(06/26/08,19:50:00.00)
      )
        STATISTIC
            REPORT
              LAYOUT(LONG)
        EXEC
```

（3）Locking Report

Locking Report 提供了 DB2 子系统中的死锁和超时等信息，是 Accounting Report 和 Statistics Report 等报告的补充。例如在 Accounting Report 报告里一般只有死锁和超时信息的统计值，但是如果需要查看详细信息，还需要通过打印 Locking Report 来进一步分析。在 Locking Report 报告中，可以看到诸如哪两个线程在进行竞争、竞争发生在什么时刻、竞争的资源是哪个、最终哪个线程获得锁（也就是得到了对资源的使用权）等。

Locking Report 作业范例如下：

```
//DB2PMRPT EXEC PGM=DB2PM
//STEPLIB DD DSN=FPE.V2R1M0.SFPELOAD,DISP=SHR
//INPUTDD DD DISP=SHR,DSN=SMFMRGSH.PLEXPT2.DB2.D090328
//       DD DISP=SHR,DSN=SMFMRGSH.PLEXPT2.DB2.D090329
//JOBSUMDD DD SYSOUT=*
//ACWORK   DD DSN=&&ACWK,DCB=(RECFM=VBS,LRECL=32756,
//          BLKSIZE=6233),
//        SPACE=(CYL,(1000,1000)),UNIT=(SYSDA,15)
//STWORK   DD DSN=&&STWK,
//         DCB=(RECFM=VBS,LRECL=32756,BLKSIZE=6233),
//        SPACE=(CYL,(1000,1000)),UNIT=(SYSDA,15)
//SORTWK01  DD   DISP=(NEW,DELETE),UNIT=SYSDA,
//        SPACE=(CYL,(500,300))
//SORTWK02  DD   DISP=(NEW,DELETE),UNIT=SYSDA,
//            SPACE=(CYL,(500,300))
//DPMOUTDD DD DUMMY
//STRPTRD  DD SYSOUT=*
//SYSIN DD *
  GLOBAL(INCLUDE(SUBSYSTEMID(PB12)),TIMEZONE(08:00)
  )
  LOCKING
    TRACE  LEVEL(LOCKOUT)
    EXEC
```

（4）Audit Report

Audit Report 主要显示 DB2 资源使用及授权等方面的信息，此类数据一般不是面向性能的，主要用于审计。

Audit Report 作业范例如下：

```
//       DD DISP=SHR,DSN=SMFMRGSH.PLEXPT2.DB2.D090329
//JOBSUMDD DD SYSOUT=*
//ACWORK   DD DSN=&&ACWK,
//         DCB=(RECFM=VBS,LRECL=32756,BLKSIZE=6233),
//         SPACE=(CYL,(1000,1000)),UNIT=(SYSDA,15)
//STWORK   DD DSN=&&STWK,
//         DCB=(RECFM=VBS,LRECL=32756,BLKSIZE=6233),
//         SPACE=(CYL,(1000,1000)),UNIT=(SYSDA,15)
//SORTWK01  DD   DISP=(NEW,DELETE),UNIT=SYSDA,
//           SPACE=(CYL,(500,300))
//SORTWK02  DD   DISP=(NEW,DELETE),UNIT=SYSDA,
//           SPACE=(CYL,(500,300))
//DPMOUTDD DD DUMMY
//STRPTRD  DD SYSOUT=*
//SYSIN DD *
  GLOBAL(INCLUDE(SUBSYSTEMID(PB12)),TIMEZONE(08:00)
  )
  AUDIT
    REPORT
    LEVEL (DETAIL)
```

6.7.1.2 查看DB2 系统LOG信息进行性能监控

另外一个监控 DB2 系统性能的方法是查看目标 DB2 的 MSTR 地址空间，通过其中的各类信息可以获得很多实用信息。在 MSTR 地址空间中，一般查看 JESMSGLG 部分来获取数据库运行的相关信息。这些信息主要包括 DB2 启动和停止时间、DB2 使用的 ZPARM 的名称、DB2 BSDS 的相关信息、DB2 Active Log 名称及 RBA 等信息、DB2 Archive Log 名称及 RBA 信息。另外，连接到 DB2 的线程，包括联机交易或批量作业，如果在访问 DB2 的过程中发生异常情况，通常也会在 MSTR 地址空间中体现，例如资源不可用等。

DB2 启动相关信息如下：

```
                  J E S 2  J O B  L O G  --  S Y S T E M  P T 1 1  --  N O

19.49.35 S0105738 ---- WEDNESDAY, 18 NOV 2009 ----
19.49.35 S0105738  IEF695I START PB11MSTR WITH JOBNAME PB11MSTR IS ASSIGNED TO USER
```

```
STCUSER , GROUP #GRPSTC
    19.49.35 S0105738  $HASP373 PB11MSTR STARTED      （启动时间）
    19.49.35 S0105738  IEF403I PB11MSTR - STARTED - TIME=19.49.35
    19.49.35 S0105738  DSNZ002I -PB11 DSNZINIT SUBSYSTEM PB11 SYSTEM PARAMETERS LOAD
MODULE NAME IS ZPRMPB11  （ZPARM 名称）
    19.49.36 S0105738  IXL014I IXLCONN REQUEST FOR STRUCTURE DSNPB01_SCA  863
      863           WAS SUCCESSFUL.  JOBNAME: PB11MSTR ASID: 00BB
      863           CONNECTOR NAME: DB2_PB11 CFNAME: BR12C1
    19.49.38 S0105738  S PB11IRLM
    19.49.41 S0105738  DSNY001I  -PB11 SUBSYSTEM STARTING
    19.49.41 S0105738  DSNJ127I  -PB11 SYSTEM TIMESTAMP FOR BSDS= 09.322 19:39:56.57(BSDS
信息)
    19.49.49 S0105738  DSNJ001I  -PB11 DSNJW007 CURRENT COPY 1 ACTIVE LOG  306
      306           DATA SET IS DSNAME=DSNPB01.PB11.LOGCOPY1.DS16,  (ACTIVE LOG 名称)
      306           STARTRBA=0B40C19E7000,ENDRBA=0B40ED906FFF   (RBA 信息)
    19.49.49 S0105738  DSNJ001I  -PB11 DSNJW007 CURRENT COPY 2 ACTIVE LOG  307
      307           DATA SET IS DSNAME=DSNPB01.PB11.LOGCOPY2.DS16,
      307           STARTRBA=0B40C19E7000,ENDRBA=0B40ED906FFF
    19.49.49 S0105738  DSNJ099I  -PB11 LOG RECORDING TO COMMENCE WITH  308
      308           STARTRBA=0B40E22D1000
    19.49.49 S0105738  S PB11DBM1
    19.49.52 S0105738  S PB11DIST
    19.49.54 S0105738  DSNR001I  -PB11 RESTART INITIATED
    19.49.54 S0105738  DSNR003I  -PB11 RESTART...PRIOR CHECKPOINT RBA=0B40E22AE1A7
    19.49.55 S0105738  DSNR004I  -PB11 RESTART...UR STATUS COUNTS  609
      609           IN COMMIT=0, INDOUBT=0, INFLIGHT=0, IN ABORT=0, POSTPONED ABO
    19.49.55 S0105738  DSNR005I  -PB11 RESTART...COUNTS AFTER FORWARD  610
      610           RECOVERY
      610           IN COMMIT=0, INDOUBT=0
    19.49.55 S0105738  DSNR006I  -PB11 RESTART...COUNTS AFTER BACKWARD  611
      611           RECOVERY
      611           INFLIGHT=0, IN ABORT=0, POSTPONED ABORT=0
    19.49.55 S0105738  DSNG007I  -PB11 DB2 CATALOG LEVEL (810) CODE LEVEL (810) MODE
    19.49.59 S0105738  DSNR002I  -PB11 RESTART COMPLETED
    19.49.59 S0105738  -PB11RECOVER POSTPONED
    19.49.59 S0105738  DSNV434I  -PB11 DSNVRP NO POSTPONED ABORT THREADS FOUND
    ……
    23.46.09 S0105738  DSNJ001I  -PB11 DSNJW307 CURRENT COPY 2 ACTIVE LOG  142
      142           DATA SET IS DSNAME=DSNPB01.PB11.LOGCOPY2.DS04,
      142           STARTRBA=0B40ED907000,ENDRBA=0B4119826FFF
    23.46.52 S0105738  DSNJ003I  -PB11 DSNJOFF3 FULL ARCHIVE LOG VOLUME  172 (ARCHIVE LOG
信息)
      172           DSNAME=DSNPB01.PB11.ARCLG1.D09322.T2346099.A0016820,
```

```
      172           STARTRBA=0B40C19E7000, ENDRBA=0B40ED906FFF,
STARTLRSN=C51AF652474F,
      172           ENDLRSN=C51B76CFCBEB, UNIT=3390, COPY1VOL=P1AR1A, VOLSPAN=00,
      172           CATLG=YES
    23.46.52 S0105738  DSNJ003I  -PB11 DSNJOFF3 FULL ARCHIVE LOG VOLUME  173
      173           DSNAME=DSNPB01.PB11.ARCLG2.D09322.T2346099.A0016820,
      173           STARTRBA=0B40C19E7000, ENDRBA=0B40ED906FFF,
STARTLRSN=C51AF652474F,
      173           ENDLRSN=C51B76CFCBEB, UNIT=3390, COPY2VOL=P2AR27, VOLSPAN=00,
      173           CATLG=YES
```

DB2 资源不可用信息如下。

```
    09.39.50 S0105738  DSNT376I  -PB11 PLAN=PLANON WITH  019
      019                 CORRELATION-ID=ENTR76300016
      019                 CONNECTION-ID=CI11PA11
      019                 LUW-ID=VTAM1.PB11LU.C52244B43E10=71375
      019                 THREAD-INFO=NGDBA:*:*:*
      019                 IS TIMED OUT. ONE HOLDER OF THE RESOURCE IS PLAN=PLANON WITH
      019                 CORRELATION-ID=ENTRMBL90093
      019                 CONNECTION-ID=CI11PA11
      019                 LUW-ID=VTAM1.PB11LU.C5224409B8D7=70162
      019                 THREAD-INFO=NGDBA:*:*:*
      019                 ON MEMBER PB11
    09.39.50 S0105738  DSNT501I  -PB11 DSNILMCL RESOURCE UNAVAILABLE  020
      020                   CORRELATION-ID=ENTR76300016
      020                   CONNECTION-ID=CI11PA11
      020                   LUW-ID=VTAM1.PB11LU.C52244B43E10=71375
      020                   REASON 00C9008E
      020                   TYPE 00000304
      020                   NAME NASEDB  .NSPATEL .X'00100248'.X'25'
```

以上信息说明两个联机交易，ID 分别为 ENTRMBL90093 和 ENTR76300016，在访问同一个表空间（NASEDB.NSPATEL）的时候，发生了死锁，最终导致超时（TIMEOUT）。ID 为 ENTR76300016 的联机交易超时失败，而 ID 为 ENTRMBL90093 的联机交易成功。

未及时 COMMIT 的作业或交易的相关信息如下：

```
    19.54.49 S0105738  DSNR035I  -PB11 DSNRPBCW WARNING - UNCOMMITTED UR  376
      376           AFTER 18 CHECKPOINTS -
      376                CORRELATION NAME = HGB02013
      376                CONNECTION ID  = BATCH
      376            LUWID = VTAM1.PB11LU.C51EF70C5A5D = 20756
      376                PLAN NAME = PLANBT
```

```
376                AUTHID = NGTNS
376                END USER ID = *
376                TRANSACTION NAME = *
376                WORKSTATION NAME = *
```

以上信息说明一个作业名称为 HGB02013 的批量作业，在经过了 18 个 CHECKPOINT 后，仍然没有进行 COMMIT，此信息为警告信息。

DB2 数据表发生数据不一致的信息如下：

```
05.22.42 S0106306  DSNI013I  -PB32 DSNIOW POTENTIALLY INCONSISTENT DATA  741
   741                    REASON 00C90207
   741                    ERQUAL 5002
   741                    TYPE 00000302
   741                   NAME BCASDB  .BSCRMAN .X'01422F33'
   741                    CONNECTION-ID=Utility
   741                    CORRELATION-ID=BP190BA
   741                    LUW-ID=VTAM1.PB32LU.C5220A63F930=0
```

以上信息说明一个运行 DB2 Utility 的作业 BP190BA 发生了数据不一致的情况，数据不一致的表空间为 BCASDB.BSCRMAN。

通过以上的信息，我们可以了解 DB2 运行的状态，获取 DB2 系统的相关配置信息，尤其是当数据库的某些线程异常，或者部分资源不可用的时候，相关信息都可以体现在 MSTR 地址空间的 LOG 中，可以帮助我们及时发现问题。

6.7.1.3　查看DB2 资源使用情况进行性能监控

DB2 的资源是否充足，是保证 DB2 系统正常运转的前提。在进行性能监控的时候，数据库管理员需要时刻关注 DB2 各类资源的使用率以保证数据库的平稳运行。在进行数据库资源监控的时候，主要使用 DISPLAY 等 DB2 的命令进行监控。监控对象主要有表空间状态、索引空间状态、Buffer Pool 状态及使用情况、GPB 状态及使用情况、DB2 ACTIVE LOG 状态及使用率、DB2 DDF 相关信息等，具体命令可以参考本书第 5 章“DB2 常用命令”。

6.7.1.4　使用EXPLAIN BIND，查看程序访问路径

DB2 性能监控还有一个方面就是监控和分析程序性能，而程序的性能在很大程度上取决于程序在数据库的访问路径。如果发现联机交易或批量作业的相关程序效率存在问题，可以通过检查程序访问路径的方法分析问题原因。

程序的访问路径可以通过查询 PLAN_TABLE 获得，但是前提是需要在程序的 BIND

或 REBIND 操作时使用 EXPLAIN(YES)参数，这样 DB2 会将访问路径相关信息写入预先定义的表 PLAN_TABLE 中。

当我们怀疑某个程序访问路径不佳的时候，可以通过如下 SQL 查看此程序在 DB2 中的访问路径：

```
SELECT * FROM qualify.PLAN_TABLE WHERE PROGRAM='XXXXXXX';
```

在查询结果中，主要关注 METHOD、ACCESSTYPE、MATCHCOLS 等几个字段。

1. METHOD

METHOD 代表连接查询表的方法，METHOD 为 0 表示直接访问，不采用连接方式，1、2、3、4 表示其他 DB2 支持的表连接查询类型。

2. ACESSTYPE

ACCESSTYPE 代表访问表的类型，一般为 R 和 I。R 代表 TABLESPACE SCAN，也就是不使用索引而是采用全表扫描的方式来进行数据读取，I 代表程序通过索引来对数据库表进行访问。通常我们认为对于大量的数据表而言，ACCESSTYPE 为 R 属于访问路径异常。

3. MTCHCOLS

MATCHCOLS 为匹配字段，当程序使用索引访问数据库表的时候，此字段才有意义，需要根据实际情况查看程序 SQL 中的谓词和索引的字段，如果应该可以匹配到的字段而没有匹配到的话，说明此程序还有优化的余地。

当我们通过 PLAN_TABLE 发现程序访问路径不佳的时候，一般需要对程序所访问的数据表进行 RUNSTATS 操作，然后对相应程序进行 REBIND，大部分访问路径不佳的问题将得到解决。

6.7.2 DB2 性能调整

在使用以上手段对 DB2 子系统进行监控后，数据库管理员需要分析所收集到的各类信息，并通过各种手段调整 DB2 系统，以提高其性能。

6.7.2.1 调整DB2 子系统配置

调整 DB2 系统配置主要包括调整 DB2 系统参数、调整 IRLM 和调整 DB2 子系统配置、调整优先级 4 个方面。

1. 调整 DB2 系统参数（ZPARM）

DB2 子系统在启动的时候需要一些参数来进行配置，这些系统参数通常被称为 DSNZPARM 或者简称 ZPARM。DSNZPARM 定义了许多与性能有关的设置，其中一些 ZPARM 会影响整个 DB2 系统的性能。对 ZPARM 的调整可以从以下几个方面展开：

（1）NUMLKTS

代表升级到表空间锁之前，单个表空间的页面或行锁的最大数量

（2）LOGLOAD

LOGLOAD 参数用于告诉 DB2 在检查点发生之前日志记录的数量，日志记录越多，重启的时间将越长。

（3）WRTHRSH

增大 WRTHRSH 的大小可以减少由于把日志信息写入日志数据集的物理 I/O 频率。

（4）IRLMRWT

IRLMRWT 控制着在超时之前等待不可用资源的时间。如果一个用户对 DB2 资源使用了锁，而另外一个用户也需要这个资源，DB2 将等待 IRLMRWT 指定的时间，如果超过这段时间还未得到相应的资源，将输入 -911 或者 -904 等信息。

（5）CTHREAD、IDFORCE、IDBACK、MAXDBAT

- CTHREAD

一个 MEMBER 上允许的最大线程数。在实际情况里，实存和虚存的大小决定了 DB2 CTHREAD 的数量。

- IDFORE

表示 DB2 允许的最大 TSO 连接数量，如果连接数量超过了定义的值，那么连接将被拒绝。

- IDBACK

表示 DB2 允许的最大的 BATCH 连接数量，如果连接数量超过了定义的值，那么连接将被拒绝。

- MAXDBAT（MAX REMOTE ACTIVE）

表示在同一时刻通过 DDF 连接到 DB2 的 ACTIVE 的线程的最大数量。

2. 调整 IRLM 参数

在启动 IRLM 时，可以在 JCL 中为 IRLM 指定几个参数，这些参数对 DB2 的性能有着重要的影响。

（1）DEADLOCK

确定 IRLM 何时执行死锁检测的周期。IRLM 必须经常检查死锁以避免对那些无法得到的资源而产生的长时间的等待。

（2）ITRACE

代表是否启动 IRLM 跟踪。由于启动此选项会大大降低 DB2 系统性能，建议设置为 NO，即不进行 IRLM 跟踪。

（3）PC

代表 IRLM 锁存放在内存中的位置。PC=NO 代表锁是存放在 ECSA 中的，可以进行直接寻址，提高锁性能。

3. 调整 DB2 子系统配置

DB2 子系统的 EDM POOL、BUFFER POOL、RID POOL、SORT POOL 等参数设置与性能有密切关系，数据库管理员需要根据 DB2 系统报表进行分析及评估并进行适当扩容，以提高 DB2 系统性能。

4. 调整 DB2 子系统优先级

从整个系统层面，可以通过合理设置 WLM 优先级别，提高 DB2 子系统地址空间的优先级别，以提高整体 DB2 的响应。业界较为通用的原则是将 IRLM 地址空间设为系统最高优先级，MSTR 和 DBM1 次之。

6.7.2.2 分析调优应用程序性能

应用程序性能调优的一个重点是调优访问路径，在 6.7.1.4 节中已经介绍了程序访问路径的查看方法，本节将介绍一下帮助 DBA 和应用程序开发人员分析调整程序访问路径最常用的一个工具——EXPLAIN 的使用方法。

1. EXPLAIN 工具介绍

应用程序的访问路径生成是在执行 DB2 BIND 或 REBIND 时，DB2 优化器查看 DB2 CATALOG 表中相关统计信息，分析每一条 SQL 语句，估算出该 SQL 每种路径的开销，并选择开销最小的路径作为执行路径。如果 CATALOG 信息不能真实反映数据量分布情况，DB2 优化器就有可能选择错误的路径。同时为了方便用户，DB2 也提供了一种方法可以帮助用户在应用程序编码结束后提前模拟 BIND 过程中访问路径计算的过程，不真正生成访问路径。这就是 EXPLAIN 工具的用途。当用户想确认应用程序的执行路径，以便于判定程序的执行效率是否和预期一样时，可以用 EXPLAIN 来对应用程序中的静态、动态 SQL

语句进行路径分析和成本估计。

2. EXPLAIN 的基本用法

当需要对某个 SQL 进行分析时，可以使用下面的语句来分析：

```
EXPLAIN ALL （SET QUERYNO=INTEGER） FOR （要分析的 SQL 语句）
```

这样 DB2 会把选用的 ACCESS PATH（访问路径）放入 yyyyy.PLAN_TABLE 中，把成本估计放入 yyyyy.DSN_STATEMNT_TABLE 中，yyyyy 表示 SQLID，一般为用户名，也可以在 EXPLAIN 前，用 SET CURRENT SQLID='YYYYY'来改变 SQLID。

当上面的步骤都执行完毕后，用户可以通过访问 PLAN_TABLE 中的信息以确认 SQL 语句的访问路径。这里需注意的是执行 EXPLAIN 的环境的 DB2 CATALOG 信息要和真实情况相近，因为 EXPLAIN 是根据 DB2 CATALOG 信息来确认路径的，因此根据数据量的变化及时执行 RUNSTAT 对于 DB2 优化器选择正确的访问路径是非常有必要的。



6.8　课后习题

1．数据库备份有哪几种常用方法？（　　）

A．COPY Utility　　B．REORG Utility

C．UNLOAD Utility　　D．LOAD Utility

2．DSN1COPY 和 RECOVER 两个 Utility 有哪些相同点？（　　）

A．都是 DB2 Utility

B．都可作为数据恢复使用

C．DB2 子系统由故障未启动的情况下不可使用

D．都可以恢复到某一个时间点

3．DB2 系统表的备份主要指哪两个 DB 里的表？（　　）

A．DSNDB01　　B．WORKDB

C．DSNDB06　　D．DSNDB04

4．检查 DB2 ACTIVE 的时候，出现以下哪些信息，需要重点关注？（　　）

A．OFFLOAD TASK IS (AVAILABLE)

B．OFFLOAD TASK IS (BUSY)

C．FULL LOGS TO OFFLOAD = 30 OF 32

D．FULL LOGS TO OFFLOAD = 0 OF 32

5．DB2 表空间处于以下哪种状态的时候是可以进行读写操作的？（ ）

A．RO B．RW C．COPY D．LPL

6．以下哪种线程状态是异常的？（ ）

A．ACTIVE B．INACTIVE

C．INDOUBT D．POSTPONED

7．以下哪种情况需要对 DB2 表进行重组？（ ）

A．DB2 文件超过 30000 TRACKS

B．DB2 表空间扩展次数过多

C．DB2 表排列顺序不整齐

D．DB2 索引空间扩展次数过多

8．对 DB2 表进行 RUNSTAT 操作时，需要注意以下哪点？（ ）

A．在所有表完成 RUNSTATS 后，尽量安排进行相关程序的 REBIND

B．数据量很少的情况下，不建议进行 RUNSTAT

C．对表和表所有的 INDEX 的 RUNSTAT 要同时进行

D．数据库表空间扩展次数过多时，需要进行 RUNSTAT

9．STOSPACE 操作可以更新哪些系统表？（ ）

A．SYSIBM.SYSINDEXES B．SYSIBM.SYSTABLESPACE

C．SYSIBM.SYSINDEXPART D．SYSIBM.SYSSTOGROUP

10．DB2 TRACE 包含以下哪几种？（ ）

A．Performance Trace B．Accounting Trace

C．Statistics Trace D．Auditing Trace

参考文献

1．萨师煊，王珊. 数据库系统概论（第三版）. 北京：高等教育出版社，2000

2．萨师煊，王珊. 数据库系统概论（第四版）. 北京：高等教育出版社，2007

3．（美）Craig S.Mullins 著，段小璐等译. DB2 开发人员指南. 北京：机械业出版社，2002

4．李志伟. DB2 基础教程. 北京：清华大学出版社，2003

5．《DB2 Universal Database for z/OS Utility Guide and Reference》

6．《DB2 for z/OS Stored Procedures：Through the CALL and Beyond》

7．《DB2 UDB for z/OS V8 Administration Guide》

8．《DB2 UDB for zOS V8 Command Reference》

9．《DB2 for z/OS：Data Sharing in a Nutshell》

10．《DB2 Universal Database for z/OS Planning and Administration》

11．《DB2 UDB for z/OS V8 Application Programming and SQL Guide》

12．《DB2 UDB for z/OS V8 Data Sharing：Planning and Administration》

13．《DB2 UDB for z/OS V8 Installation Guide》

14．《DB2 UDB for z/OS V8 Application Programming Guide and Reference》

15．《DB2 UDB for z/OS V8 Messages and Codes》

16．《DB2 UDB for z/OS V8 Release Planning Guide》

17．《DB2 UDB for z/OS V8 SQL Reference》

18.《DB2 UDB for z/OS V8 Reference for Remote DRDA Requesters and Servers》
19.《DB2 UDB for z/OS V8 Utility Guide and Reference》
20.《DB2 UDB for z/OS V8 What’s New?》
21.《DB2 Administration Tool User’s Guide》
22.《DB2 UDB for z/OS V8 RACF Access Control Module Guide》

附录A DB2 Admin Tool 简介

DB2 Administration Tool 是主机平台管理用户数据库对象的工具，它能够简化数据库管理员的日常管理工作。DB2 Administration Tool 使浏览数据库 Catalog 信息变得简单方便，并以更加可读的方式呈现给用户。DB2 Administration Tool 能够在数据库对象之间方便地切换。并且 DB2 Administration Tool 集成了许多 Utilities 及管理工具，能够对数据库对象进行各项管理工作。

在 z/OS 系统中，用户可以通过 ISPF 面板中预先裁剪好的选项进入 DB2 Admin Tool，如图 A.1 所示。

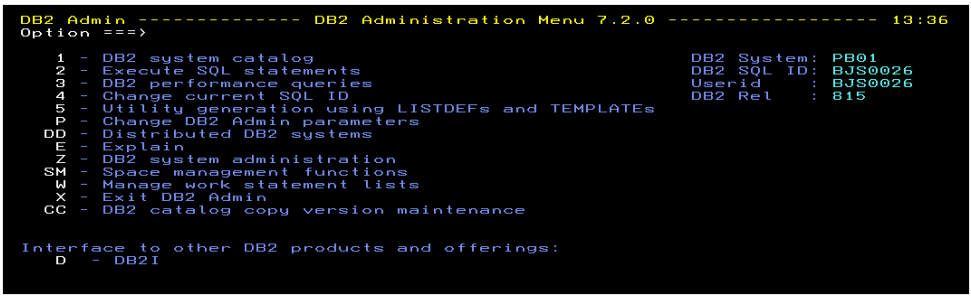


图 A.1　通过预先裁剪好的选项进入 DB2 Admin Tool

下面分别介绍 DB2 Admin Tool 的主要功能。

1．显示Catalog表相关信息

DB2 Admin 提供了对于 DB2 Catalog 中相关对象逐层的浏览功能，随时能够逐层深入地查看并解释所关心对象的相关信息，将数据库 Catalog 信息以更加易读的方式提供给客户。

2．执行动态SQL语句

在 DB2 Admin 中可以从屏幕或者 Data Set 中提交动态 SQL 执行，还可以通过命令行交互地生成并执行 SQL SELECT 语句。另外，通过从面板输入参数，在 DB2 Admin 中还可以执行 GRANT，REVOKE，CREATE，DROP，LABEL ON 和 COMMENT ON 等语句，这样即使用户不是非常清楚 SQL 语法，也可以完成相应语句的执行。DB2 Admin 还提供了对需要输入 SQL 参数的相关帮助。

3．对数据库和表空间执行DB2 命令

使用 DB2 Admin 可以针对任意数据库或表空间执行相关 DB2 命令。DB2 命令被传递给 IFI（Instrumentation facility interface），返回结果将显示在 ISPF 中。

4．运行DB2 Utility

通过 DB2 Admin 可以生成 DB2 Utility 相关的 JCL，然后提交作业执行它们。该功能可应用于针对 Storage Group、表空间、表和索引的 Utility。例如用户可以生成执行针对某表空间的 COPY、REORG、RUNSTATS 的 Utility 的 JCL，生成的 JCL 将包含一个 JOB statement、EXEC statement 及所有所需的 DD 参数。当 JCL 生成好后，DB2 Admin 调用 ISPF 编辑器使用户可以修改，提交作业或者将其复制到另外一个 Data Set 中去。

5．执行复杂查询

用户可以查询性能和空间的使用情况，并借此判断是否需要：执行 RUNSTATS 或 STOSPACE Utility；重组或重新设计数据库或索引；修改表的锁定规则；删除索引；将某表从其所在表空间中分离出来；增加表空间或索引空间的首分空间大小；缩小表空间占用的磁盘空间等。

6．使用EXPLAIN功能

DB2 Admin 工具中的 EXPLAIN 模块提供了 DB2 的 EXPLAIN 功能的相关支持

（EXPLAIN 可以用来收集 DB2 对于查询的访问路径的相关信息）。

7．管理SQL ID

用户可以通过输入一个新的或者从 Secondary SQL ID 列表中选择一个来改变当前的 DB2 SQL ID。DB2 Admin 工具可以显示出允许使用的 SQL ID 的列表。

8．执行系统管理功能

使用 DB2 Admin 工具可以完成以下一些系统管理功能：显示线程；显示和终止 Utility；显示和管理跟踪器；显示和修改 RLIMITS；显示和修改 Buffer Pool；显示和设定 Archive Log 的参数，以及将日志 Archive 出去；显示 DB2 的系统参数和修改动态参数；对于 DDF（Distributed Data Facility）还可以启下 DDF，显示和修改 CDB（Communication Database）；显示和终止远程连接线程等。

9．逆向处理DB2 对象

Reverse engineering 可以生成用于重建 DB2 对象的 SQL。通过使用 Reverse engineering（配合表的 UNLOAD 和 LOAD），数据库对象可以移动到其他地方，所使用的 SQL 语句可以联机或者通过批量作业生成。

10．DB2 的预设管理功能

DB2 Admin 可以显示、添加、修改或删除资源限定表（Resource Limit Table）中的预设管理记录（Predictive Governing Rows）。除此之外，当 DB2 Admin 运行某动态 SQL 语句收到预设管理警告（SQLCODE 为+495）后，DB2 Admin 会询问用户是否继续执行还是取消。如果判断出执行 SQL 语句会超过错误界限（SQLCODE 为-495），DB2 Admin 会显示一条报错信息，而该 SQL 语句也不会执行。用户可以使用预设管理界限来防止对于 Catalog 表的有害操作。

11．修改DB2 表定义

可以通过 DB2 Admin 来修改表的定义，允许的修改包括：修改数据库，表空间、所有者、以及表名；修改数据库表中列的定义；修改表中列的排列顺序；插入或删除列。

12．迁移DB2 数据至其他DB2 系统

可以通过 DB2 Admin 将某 DB2 系统中的数据复制至其他 DB2 系统。当需要新建一个

独立的 DB2 测试系统或者将测试系统迁移至生产环境时，该功能非常实用。用户还可以使用该功能来整合数据库。

13．执行空间管理功能

DB2 Admin 还可以实现对于诸如修改 Page Set 大小，移动 STOGROUP 和 VCAT 中的 Page Set，预估新建表空间和索引的分配空间等空间相关的操作。

附录 B DB2 PM简介

DB2 PM 作为 DB2 子系统性能分析工具，可按在线和报告的形式显示缓冲池数据集的统计数据，从而有效准确地帮助 DBA 评价 DB2 系统的性能和应用程序的性能，帮助找出各种性能问题，提供 DBA 系统优化和应用优化的依据和方法。它还提供关于暂停和计数的更为详细的信息，以在扩展的动态 SQL 语句缓存内动态执行 SQL 语句。

下面将分别介绍联机分析和批量分析。

1．Online Monitor分析

DB2 ONLINE MONITOR 将 DB2 性能以一种简而易懂的形式展现给用户。在 z/OS 系统中，用户可以通过 ISPF 面板中预先裁剪好的对应选项进入 Performance Monitor，如图 B.1 所示。

DB2 系统会生成有关性能的数据，但并不提供任何分析数据的工具。实时监视器（Online Monitor）使得用户可以查看某个活动的 DB2 子系统的性能和问题情况。

实时监视器以汇总形式显示出 DB2 性能信息，以便于理解和分析。

用户可以使用实时监视器了解整个 DB2 系统的性能和效率，查看某个应用的性能和资源使用情况，评估某个应用对于系统和其他应用的影响，帮助分析改进 SQL 语句，检查潜在问题，确定 DB2 调节需求。

当某个应用或 DB2 子系统有改变时，实时监视器可以帮助分析其影响，这对于判断改变是否有利于提升性能非常有用。

当用户对 DB2 性能不太满意时，实时监视器可以帮助用户判断修改哪些环节以优化 DB2 性能。实时监视器可以记录 DB2 行为和事件，并将这些信息提供给用户以分析潜在问

题。用户还可以执行一个线程分析来查看某线程的性能情况及优化建议。

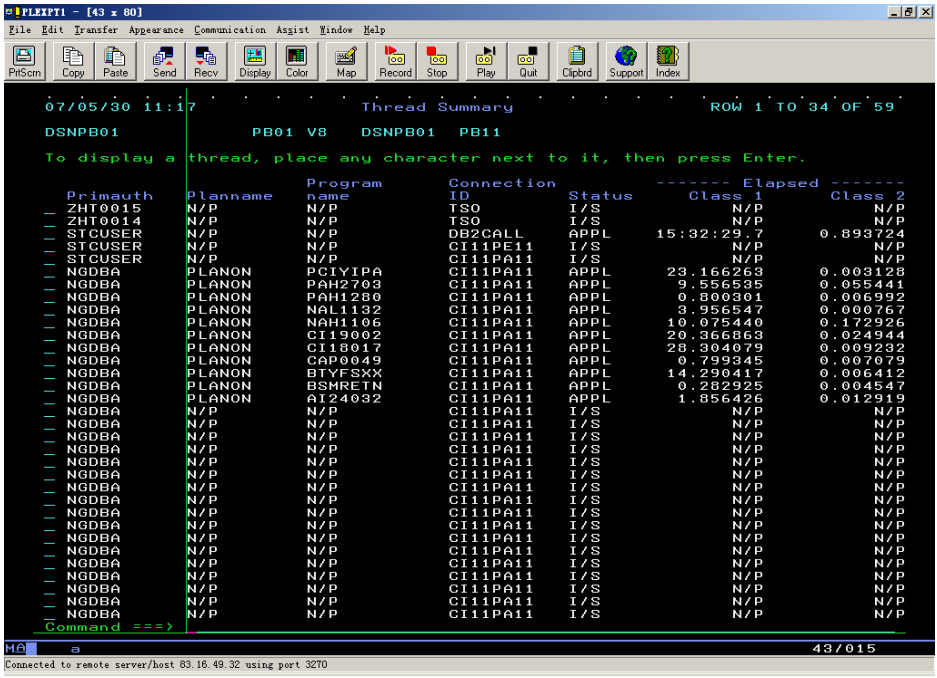


图 B.1　通过预先裁剪好的对应选项进入 Performance Monitor

2．Batch Performance Report分析

（1）ACCOUNTING LONG REPORT 分析

ACCOUNTING 报表的主要用途如下：

- 汇总特定 DB2 PLAN 的本地和异地 DB2 activity 信息。
- 汇总使用 CP 并行或 SYSPLEX 并行查询的复杂 DB2 activity 信息。
- 汇总特定 DB2 package 和 DBRM 执行的 DB2 activity 信息。
- 检测 DB2 应用程序的潜在问题。
- 根据 DB2 PM identifiers 例如 location，authorization ID 或 plan name，追踪 DB2 资源的使用情况，常用于趋势分析。
- 指明不符合用户定义规则而失败的 DB2 线程。

（2）STATISTICS LONG REPORT 分析

Statistics 报表汇总了各个方面系统级的性能数据，包括时间消耗、SQL activity 的相关

信息、锁信息、缓冲池使用情况等。

STATISTICS 报表的主要用途如下：

- 显示 DB2 关键部件的系统级统计信息。
- 比对两个或多个报表时间段（reporting intervals）的系统性能。
- 评估每个 DB2 子系统的系统级性能。
- 评估 DB2 Data Sharing 环境下 GROUP 的性能。
- 在一份报表中汇总系统信息。

附录C　SPUFI简介

SPUFI（SQL Processing Using File Input）是 DB2 for z/OS 中最常用的执行动态 SQL 的工具。

在 z/OS 系统中，用户可以通过 ISPF 面板中预先裁剪好的选项进入 DB2I 交付界面，选择其中的 SPUFI 选项进入 SPUFI 查询工具。SPUFI 使用 Data Set 作为 SQL 语句的输入和输出，执行查询前需先通过 DB2I 中的 D 选项设定好所访问的 DB2 Group 名，如图 C.1 所示。

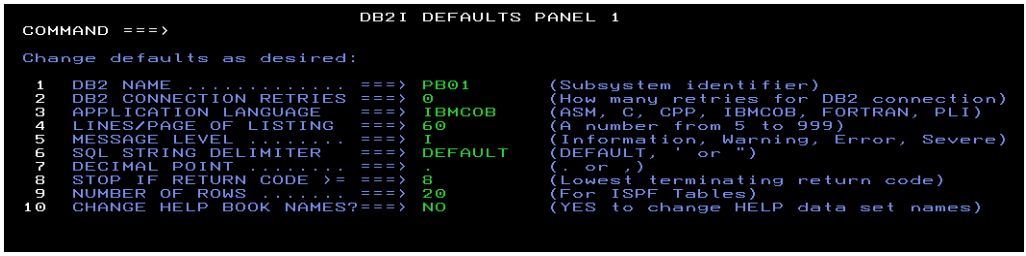


图 C.1　设定 DB2 Group 名

之后选择选项 1-SPUFI 进入作为 SQL 语句输入的 Data set 的编辑界面，编辑好后按 F3 键自动保存退出，回车后系统会将 SQL 语句的执行结果显示出来，如图 C.2 所示。

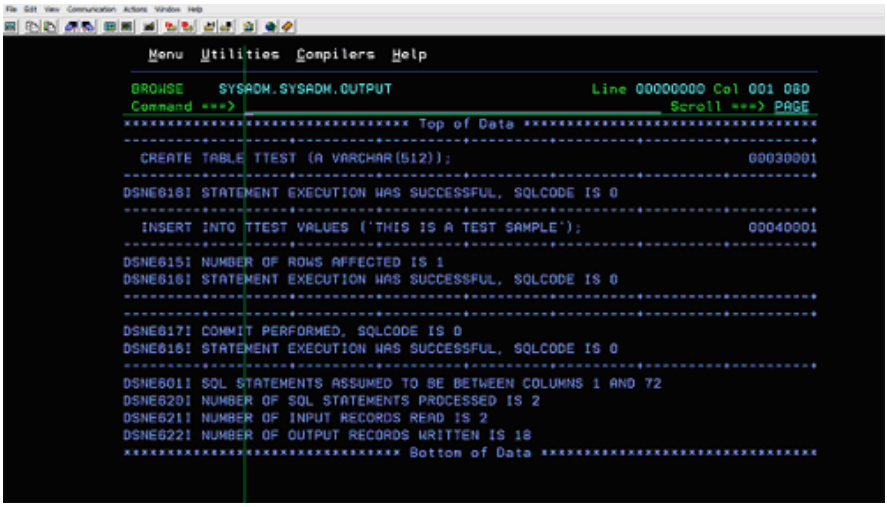

Menu Utilities Compilers Help
BROWSE SYSADM.SYSADM.OUTPUT Line 00000000 Col 001 080
Command ===> Scroll ===> PAGE
****** Top of Data ******
CREATE TABLE TTEST (A VARCHAR(512)); 00030001
DSNE618I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
INSERT INTO TTEST VALUES ('THIS IS A TEST SAMPLE'); 00040001
DSNE615I NUMBER OF ROWS AFFECTED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 2
DSNE621I NUMBER OF INPUT RECORDS READ IS 2
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 18
****** Bottom of Data ******

图 C.2　执行结果

附录 D 参考答案

第 1 章参考答案

1．答案：

文件系统和数据库系统一样，都可以用于长期保存数据，并支持数据的查询、修改、插入和删除等操作，同时文件系统也实现了记录内的结构性，使应用程序与数据之间有了一定的独立性。但文件系统较数据库系统来说，其数据共享性很差，冗余度也很大，这是由文件系统的特点所决定的。另外，文件系统的数据独立性也很差，应用系统不易扩展。而在数据库系统中，数据不再针对某一应用，而是面向全组织，实现了整体数据的结构化，使得数据库系统具备了共享性高、冗余度强、易扩展的良好特性。同时数据库系统具有很高的数据独立性，所有数据由 DBMS 统一管理和控制，保证了数据的安全性、完整性以及并发控制和故障恢复。

2．答案：

层次模型的优点有：（1）层次数据模型本身比较简单；（2）对于实体间联系是固定的，且预先定义好的应用系统，采用层次模型来实现，其性能优于关系模型，不低于网状模型；（3）层次数据模型提供了良好的完整性支持。

层次模型的缺点有：（1）现实世界中很多联系是非层次性的，层次模型表示这类联系的方法很笨拙，只能通过引入冗余数据（易产生不一致性）或创建非自然的数据组织（引入虚拟结点）来解决；（2）对插入和删除操作的限制比较多；（3）查询子女结点必须通过

双亲结点；（4）由于结构严密，层次命令趋于程序化。

网状数据模型的优点有：（1）能够更为直接的描述现实世界，如一个结点可以有多个双亲；（2）具有良好的性能，存取效率较高。

网状数据模型的缺点有：（1）结构比较复杂，而且随着应用环境的扩大，数据库的结构就变得越来越复杂，不利于最终用户掌握；（2）其 DDL，DML 语言复杂，用户不容易使用。由于记录之间联系是通过存取路径实现的，应用程序在访问数据时必须选择适当的存取路径，因此，用户必须了解系统结构的细节，加重了编写应用程序的负担。

3．答案：

关系模型的完整性规则包括实体完整性、参照完整性和用户定义完整性。

（1）实体完整性：关系的主属性不能为空值。空值（null）就是指不知道或是不能使用的值，注意它与数值 0 和空字符串的意义是不一样的。

（2）参照完整性：如果关系 R1 的外码与关系 R2 的主码相符，那么关系 R1 的外码的每个值必须在关系 R2 的主键的值中找到或者是空值。

（3）用户定义完整性：是针对某一具体的实际数据库的约束条件。它由应用环境所决定，反映某一具体应用所涉及的数据必须满足的要求。关系模型提供定义和检验这类完整性的机制，以便用统一的、系统的方法处理，而不必由应用程序承担这一功能。

4．答案：

DML（DATA MANPULATION LANGUAGE）—数据操作语言

主要 SQL 语句有：SELECT，DELETE，UPDATE，INSERT 等。

DDL（DATA DEFINITION LANGUAGE）—数据定义语言

主要 SQL 语句有：CREATE，ALTER，DROP，DECLARE 等。

DCL（DATA CONTROL LANGUAGE）—数据控制语言

主要 SQL 语句有：GRANT，REVOKE，COMMIT，ROLLBACK 等。

5．答案：

错误，在数据库中 NULL 表示在一个字段中没有数据，它与该字段中数据为零或为空不是同一个概念，在进行赋值或比较时，数据库将返回 Unknown 这种不正常状态。DB2 中为了测试某个字段是否为 NULL，需要使用 IS 操作进行。

6．答案：

连接查询是通过连接操作从多个表中选择数据的一种查询，而子查询则是把查询结果作为参数返回给另一个查询的一种查询。

如果两者在逻辑上等价，连接查询的效率更好些，因为如果采用子查询方式，为确保消除重复值，必须为外部查询的每个结果都进行嵌套查询。

7．答案：

静态 SQL 指的是程序中写定的 SQL 代码，这些代码在程序运行时不能被更新，这类 SQL 在运行之前经过了预编译处理；而动态 SQL 则是在程序运行时构建 SQL 并把查询语句提交给数据库，数据库运行完成后再将结果返回给程序变量。

静态 SQL 可提高运行速度且经过了编译错误检查，但其灵活性较差，需要编写更多的代码，而动态 SQL 则与之相反，其灵活性高，但运行效率则较静态 SQL 有所降低。

8．答案：

```
CREATE TRIGGER T_TBLA
   AFTER INSERT ON TBLA
   FOR EACH ROW MODE DB2SQL
   BEGIN ATOMIC
   UPDATE TBLB SET COLA = COLA + 1, COLB = CURRENT TIMESTAMP;
  END
```

9．答案：

需求分析阶段的目标是通过详细调查现实世界要处理的对象（组织、部门、企业等），充分了解原系统（手工系统或计算机系统）工作概况，明确用户的各种需求，然后在此基础上确定新系统的功能。新系统必须充分考虑今后可能的扩充和改变，不能仅仅按当前应用需求来设计数据库。

需求分析阶段需要调查的内容包括调查、收集与分析用户在数据管理中的信息要求、处理要求、安全性与完整性要求。

10．答案：

E-R 图即“实体-关系模型”（E-R 模型），是用户和数据库设计人员之间进行交流的工具，在设计数据库系统之前，需要使用 E-R 图将现实世界中的实体和实体之间的联系转换为概念模型。构成 E-R 模型的基本元素是：实体、属性和联系。

第 2 章参考答案

1．D　2．A　3．A　4．B　5．C　6．D
7．True　8．A　9．True　10．D、E　11．C　12．False　13．B
14．D　15．A

第 3 章参考答案

1．答案：

DB2 Data Sharing 技术的主要优点是：提高 DB2 数据的可用性；提高 DB2 系统的处理能力；更加灵活的 DB2 系统环境配置；提升交易处理能力。

2．答案：

Parallel Sysplex 系统主要由 3 个组件构成：IBM z/os 主机、时钟控制器（Sysplex Timer）和 CFs 耦合体。

3．答案：

DB2 Data Sharing 通过以下两点，一是通过并发访问控制（Global Locking，即全局锁机制的控制）来保证并发性；二是通过对修改过的数据的一致性控制（Managing Changed Data）来控制数据一致性。

4．答案：

在 DB2 Data Sharing 中某个 DB2 MEMBER 的 Active Log 文件写满的时候，DB2 会自动将此 Active Log 文件中的内容复制到相应的 Archive Log 文件中。

5．答案：

在 DB2 Data Sharing 中，提供了一种特殊的 DB2 MEMBER 的启动模式“Light”。使用此模式启动 DB2 后，此 DB2 MEMBER 会在启动释放保留锁（Retained Lock）后，自动下宕。

第 4 章参考答案

1．答案：

COPY Utility，相对 UNLOAD Utility 要快一些，因为 UNLOAD Utility 的作用单位是

ROW，COPY Utility 作用单位是 PAGE。

2．答案：

REORG TABLESPACE Utility 对 TABLE SPACE 进行重组，以提升访问 TABLESPACE 的性能，并回收碎片。在重组时，可以通过指定 STATISTICS 参数，进行收集与访问路径相关的数据，存放在 CATALOG 表中。

3．答案：

RUNSTATS Utility 用来收集 TABLESPACE，INDEX，PARTITION， TABLE 的相关信息，并存储在 DB2 CATALOG 表中。RUNSTATS 有两种形式：RUNSTATS TABLESPACE 和 RUNSTATS INDEX。RUNSTATS TABLESPACE 可以获取包括 TABLESPACE，INSEX，COLUMNS 的信息；而 RUNSTATS INDEX 只获取 INDEX 的相关信息。

4．答案：

RBDP：PARTITIONING INDEX 的 PHYSICAL 或 LOGICAL PARTITION 处于 REBUILD-PENDING 状态。只有单独的 INDEX PARTITION 不可被访问。通过 REBUILD 这些单独的 PARITITION 可以 RESET REBUILD-PENDING 状态。

RBDP*：NON-PARTITIONING INDEX 处于 REBUILD-PENDING 状态。整个 INDEX 都无法访问。只要 REBUILD 受影响的 LOGICAL PARTITION 就可以 RESET REBUILD-PEDING 状态。

PSRBD：NON-PARTITIONING INDEX SPACE 处于 REBUILD-PENDING 状态。整个 IDNEX SPACE 不可用。需要对整个 IDNEX 做 REBUILD。

5．答案：

当修改一个表结构，比如 ALTER TABLE ADD COLUMN 后，必须运行 REORG 作业才能使用 DSN1COPY。

使用 DSN1COPY 而不指定 OBIDXLAT 可能会导致 OBID 的不可用。例如在下列的情况下：

- 当用户用 DSN1COPY 创建数据后，在使用数据之前进行表的删除和重建；
- 当源 DB2 子系统和目标 DB2 子系统有下列差异时：
 - — TABLESPACE 的 BUFFERPOOL 和 NUMPARTS 参数。
 - — TABLE 中除了 NAME，TABLESPACE NAME 和 DATABASE NAME 的参数。

— 用户在源 DB2 子系统和目标 DB2 子系统进行定义和删除这些 TABLESPACE，INDEX，TABLE 的顺序。

- 当使用 DSN1COPY 将 IMAGE COPY 文件恢复到一个 DB2 文件时，按如下方式指定 SYSUT2 文件：
 — 如果 SYSUT1 为单个 PARTITION 的 IMAGECOPY，SYSUT2 必须为 TABLESPACE 的该 PARTITION 的文件名。并指定 NUMPARTS（nn）参数，nn 为 TABLESPACE 的 PARTITION 的总数。
 — 如果 SYSUT1 为 TABLESPACE 所有 PARTITION 的 IMAGECOPY，SYSUT2 必须为 TABLESPACE 的第一个 PARTITION 的文件名。并指定 NUMPARTS（nn）参数，nn 为 TABLESPACE 的 PARTITION 的总数。
 — 如果 SYSUT1 为一个有多文件的 LINEAR TABLESPACE 的某个文件的 IMAGECOPY，SYSUT2 必须为那个输出的文件名，不需要指定 NUMPARTS 参数。
 — 如果 SYSUT1 为一个有多文件的 LINEAR TABLESPACE 的所有文件的 IMAGECOPY，SYSUT2 必须为该 TABLESPACE 的第一个文件名。

6．答案：

有两个命令可用于检测和控制 DB2 Utility：

```
DISPLAY
TERMINATE
```

7．答案：

SYSUTILX 表是在 CHECHPIONT/RESTART 信息量超过 SYSUTIL 表的可用空间时保存溢出信息。

第 5 章参考答案

一．选择题：

1．A　2．C　3．D　4．A

二．填空题：

```
ACTION（RETAIN）
DISPLAY LOG
DISPLAY Utility（*）
```

三．简答题：

1 答：CHECK-pending，COPY-pending，RECOVER-pending，REORG-pending（REORP）

2 答：REBUILD-pending（RBDP），

3 答：可能因为当前 DB2 系统设置的端口号和同 LPART 上的别的 DB2 系统的端口号重复导致获取不到 IP 地址。可以重新修改 DB2 安装文件中的端口号，然后重新编译 ZPARM，重启 DB2。

第 6 章参考答案

1．A C　　2．AB　　3．AC　　4．BC　　5．B

6．CD　　7．BCD　　8．ABC　　9．ABCD　　10．ABCD

附录E 常见主机资料缩语表

EDM：Environmental Descriptor Manager

DBD：Database Descriptor

SKCT：Skeleton Cursor

SKPT：Skeleton Package

BSDS：Bootstrap Data Set

BP：Buffer Pool

RID：Record Identifier

SQL：Structured Query Language

DBMS：Database Management System（数据库管理系统）

DCL：Data Control Language（数据控制语言）

DDL：Data Definition Language（数据定义语言）

DML：Data Manipulation Language（数据）

RDBMS：Relationship Database Management System（数据库管理系统）

NF/nNf：Normal Form

E-R Model：Entity-Relation Model（实体-关系模型）

DBA：Database Administrator（数据库管理员）

DBD：Database Descriptor（数据库描述符）

BSDS：Bootstrap Data Set

MRO：Multi-region Operation
CAF：Call Attachment Facility
RRS：Resource Recovery Service
DRDA：Distributed Relational Database Architecture（分布式关系数据库体系结构）
RACF：Resource Access Control Facility
SMS：Storage Management Subsystem
WLM：Workload Manager
UOW：Unit Of Work（工作单元）
CS：Cursor stability（游标稳定性）
RR：Repeatable read（可重复读）
RS：Read stability（读稳定性）
UR：Uncommitted read（未提交读，或“脏读”）
S lock：Share Lock（共享锁）
X lock：Exclusive Lock（排他锁）
DBRM：Database Request Module
DB2I：DB2 Interface（DB2 交互面板）
UDF：User Defined Function（用户编写函数）
PM：Performance Monitor（性能监控）
Sysplex Systems Complex
CF：Coupling Facility
GBP：Group Buffer Pool（全局缓存区）
LBP：Local Buffer Pool（本地缓存区）
CFCC：Coupling Facility Control Code
ARM：Automatic Restart Manager（系统自动重启工具）
CFRM：Coupling Facility Resource Management
SSID Subsystem ID
LRSN：Log Record Sequence Number
GRECP：Group Buffer Pool Recovery Pending
LPL：Logical Page List

CFRM：Coupling Facility Resource Management

SFM：Sysplex Failure Management

ARM：Automatic Restart Manager

PM：Performance Monitor

PE：Performance Expert

DDF：Distributed Data Facility

CDB：Communication Database

SPUFI：SQL Processing Using File Input

RBDP：REBUILD-pending

反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可，复制、销售或通过信息网络传播本作品的行为；歪曲、篡改、剽窃本作品的行为，均违反《中华人民共和国著作权法》，其行为人应承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。

为了维护市场秩序，保护权利人的合法权益，我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为，本社将奖励举报有功人员，并保证举报人的信息不被泄露。

举报电话：（010）88254396；（010）88258888
传　　真：（010）88254397
E-mail：　dbqq@phei.com.cn
通信地址：北京市万寿路 173 信箱
　　电子工业出版社总编办公室
邮　　编：100036

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
Broadview®
www.broadview.com.cn

《大型主机 DB2 数据库基础教程》读者交流区

尊敬的读者：

感谢您选择我们出版的图书，您的支持与信任是我们持续上升的动力。为了使您能通过本书更透彻地了解相关领域，更深入的学习相关技术，我们将特别为您提供一系列后续的服务，包括：

1. **提供本书的修订和升级内容、相关配套资料；**
2. **本书作者的见面会信息或网络视频的沟通活动；**
3. **相关领域的培训优惠等。**

请您抽出宝贵的时间将您的个人信息和需求反馈给我们，以便我们及时与您取得联系。

您可以任意选择以下三种方式与我们联系，我们都将记录和保存您的信息，并给您提供不定期的信息反馈。

1．短信

您只需编写如下短信：B11058+您的需求+您的建议

发送到1066 6666 789（本服务免费，短信资费按照相应电信运营商正常标准收取，无其他信息收费）

为保证我们对您的服务质量，如果您在发送短信24小时后，尚未收到我们的回复信息，请直接拨打电话（010）88254369。

2．电子邮件

您可以发邮件至jsj@phei.com.cn或editor@broadview.com.cn。

3．信件

您可以写信至如下地址：北京万寿路173信箱博文视点，邮编：100036。

如果您选择第2种或第3种方式，您还可以告诉我们更多有关您个人的情况，及您对本书的意见、评论等，内容可以包括：

（1）您的姓名、职业、您关注的领域、您的电话、E-mail地址或通信地址；

（2）您了解新书信息的途径、影响您购买图书的因素；

（3）您对本书的意见、您读过的同领域的图书、您还希望增加的图书、您希望参加的培训等。

如果您在后期想退出读者俱乐部，停止接收后续资讯，只需发送“B11058+退订”至10666666789即可，或者编写邮件“B11058+退订+手机号码+需退订的邮箱地址”发送至邮箱：market@broadview.com.cn 亦可取消该项服务。

同时，我们非常欢迎您为本书撰写书评，将您的切身感受变成文字与广大书友共享。我们将挑选特别优秀的作品转载在我们的网站（www.broadview.com.cn）上，或推荐至CSDN.NET等专业网站上发表，被发表的书评的作者将获得价值50元的博文视点图书奖励。

我们期待您的消息！

博文视点愿与所有爱书的人一起，共同学习，共同进步！

通信地址：北京万寿路 173 信箱　博文视点（100036）　　电话：010-51260888

E-mail：jsj@phei.com.cn，editor@broadview.com.cn